Available from the IACR Cryptology ePrint Archive as Report 2007/xxx.

Non-Interactive Anonymous Credentials

Mira Belenkiy
Brown University
mira@cs.brown.edu

Melissa Chase
Brown University
mchase@cs.brown.edu

Markulf Kohlweiss
K.U. Leuven
mkohlwei@esat.kuleuven.be

Anna Lysyanskaya
Brown University
anna@cs.brown.edu

September 29, 2007

Abstract

In this paper, we introduce P-signatures. A P-signature scheme consists of a signature scheme, a commitment scheme, and (1) an interactive protocol for obtaining a signature on a committed value; (2) a *non-interactive* proof system for proving that the contents of a commitment has been signed; (3) a non-interactive proof system for proving that a pair of commitments are commitments to the same value. We give a definition of security for P-signatures and show how they can be realized under appropriate assumptions about groups with bilinear map. Namely, we make extensive use of the powerful suite of non-interactive proof techniques due to Groth and Sahai.

Our P-signatures enable, for the first time, the design of a practical non-interactive anonymous credential system whose security does not rely on the random oracle model. In addition, they may serve as a useful building block for other privacy-preserving authentication mechanisms.

1 Introduction

Anonymous credentials [Cha85, Dam90, Bra99, LRSW99, CL01, CL02, CL04] let Alice prove to Bob that Carol has given her a certificate. Anonymity means that Bob and Carol cannot link Alice's request for a certificate to Alice's proof that she possesses a certificate. In addition, if Alice proves possession of a certificate multiple times, these proofs cannot be linked to each other. Anonymous credentials are an example of a privacy-preserving authentication mechanism, which is an important theme in modern cryptographic research. Other examples include group signatures [CvH91, CS97, ACJT00, BBS04a, BW06, BW07a], electronic cash [CFN90, FY93, CP93, Bra93, EJA$^+$04, CHL05, Wei05, CHL06, CGH06], and anonymous authentication [TFS04, DDP05, NSN05, TS06, CHK$^+$06]. In a series of papers, Camenisch and Lysyanskaya [CL01, CL02, CL04] identified a key building block commonly called "a CL-signature". A CL-signature is a signature scheme with a pair of useful protocols.

The first protocol, called *Issue*, lets a user obtain a signature on a committed message without revealing the message. The user wishes to obtain a signature on a value x from a signer with public key pk. The user forms a commitment $comm$ to value x and gives $comm$ to the signer. After running the protocol, the user obtains a signature on x, and the signer learns no information about x other than the fact that he has signed the value that the user has committed to.

The second protocol, called *Prove*, is a zero-knowledge proof of knowledge of a signature on a committed value. The prover has a message-signature pair $(x, \sigma_{pk}(x))$. The prover has obtained it by either running the Issue protocol, or by querying the signer on x. The prover also has a commitment $comm$ to x. The verifier also knows $comm$. The prover proves in zero-knowledge that he knows a pair (x, σ) and a value $opening$ such that $\mathsf{VerifySig}(pk, x, \sigma) = \mathsf{accept}$ and $comm = \mathsf{Commit}(x, opening)$.

It is clear that using general secure two-party computation [Yao86] and zero-knowledge proofs of knowledge of a witness for any NP statement [GMW86], we can construct the Issue and Prove protocols from any signature scheme and commitment scheme. Camenisch and Lysyanskaya's contribution was to construct specially designed signature schemes that, combined with Pedersen [Ped92] and Fujisaki-Okamoto [FO98] commitments, allowed them to construct Issue and Prove protocols that are efficient enough to use in practice. CL-signatures have been implemented and standardized [CVH02, BCC04]. They have also been used as a building block in many other constructions [JS04, EJA$^+$04, CHL05, CHL06, DDP06, CHK$^+$06, TS06, CGH06, CLM07].

A shortcoming of the CL signature schemes is that the Prove protocol is interactive. Rounds of interaction are a valuable resource. In certain contexts, proofs need to be verified by third parties who are not present during the interaction. For example, in off-line e-cash, a merchant accepts an e-coin from a buyer and later deposits the e-coin to the bank. The bank must be able to verify that the e-coin is valid.

There are two known techniques for making the CL Prove protocols non-interactive. We can use the Fiat-Shamir heuristic [FS87], which requires the random-oracle model. A series of papers [CGH04, DNRS03, GT03] show that proofs of security in the random-oracle model do not imply security. The other option is to use general techniques: any statement in NP can be proven in non-interactive zero-knowledge [BFM88, DSMP88, BDMP91]. This option is prohibitively expensive.

In this paper we give the first *practical* non-interactive zero-knowledge proof of knowledge of a signature on a committed message. We have two constructions using two different practical signature schemes and a special class of commitments due to Groth and Sahai [GS07]. Our constructions are secure in the common reference string model.

Due to the fact that these protocols are so useful for a variety of applications, it is important to give a careful treatment of the security guarantees they should provide. In this paper, we introduce the concept of P-signatures — signatures with efficient **P**rotocols, and give a definition of security. The main difference between P-signatures and CL-signatures is that P-signatures have non-interactive proof protocols. (Our definition can be extended to encompass CL signatures as well.)

OUR CONTRIBUTIONS. Our main contribution is the formal definition of a P-signature scheme and two efficient constructions.

Anonymous credentials are an immediate consequence of P-signatures (and of CL-signatures [Lys02]). Let us explain why (see Appendix A for an in-depth treatment). Suppose there is a public-key infrastructure that lets each user register a public key. Alice registers unlinkable pseudonym A_B and A_C with Bob and Carol. A_B and A_C are commitments to her secret key, and so they are unlinkable by the security properties of the commitment scheme.

Suppose Alice wishes to obtain a certificate from Carol and show it to Bob. Alice goes to Carol and identifies herself as the owner of pseudonym A_C. They run the P-signature Issue protocol as a result of which Alice gets Carol's signature on her secret key. Now Alice uses the P-signature Prove protocol to construct a non-interactive proof that she has Carol's signature on the opening of A_B.

Our techniques may be of independent interest. Typically, a proof of knowledge π of a witness x to a statement s implies that there exists an efficient algorithm that can extract a value x' from π such that x' satisfies the statement s. Our work uses Groth-Sahai non-interactive proofs of knowledge [GS07] from which we can only extract $f(x)$ where f is a one-way function. We formalize the notion of an f-extractable proof of knowledge and develop useful notation for describing f-extractable proofs that committed values have certain properties. Our notation has helped us understand how to work with the GS proof system and it may encourage others to use the wealth of this powerful building block.

TECHNICAL ROADMAP. We use Groth and Sahai's f-extractable non-interactive proofs of knowledge [GS07] to build P-signatures. Groth and Sahai give three instantiations for their proof system, using SXDH, DLIN, and SDA assumptions. We can use either of the first two instantiations. The SDA-based instantiation does not give us the necessary extraction properties.

Another issue we confront is that Groth-Sahai proofs are f-extractable and not fully extractable. Suppose we construct a proof whose witness x contains $a \in Z_p$ and the opening of a commitment to a. For this commitment, we can only extract $b^a \in f(x)$ from the proof, for some base b. Note that the proof can be about multiple committed values. Thus, if we construct a proof of knowledge of (msg, σ) where $msg \in Z_p$ and $\mathsf{VerifySig}(pk, msg, \sigma) = \mathsf{accept}$, we can only extract some function $F(msg)$ from the proof. However, even if it is impossible to forge (msg, σ) pairs, it might be possible to forge $(F(msg), \sigma)$ pairs. Therefore, for our proof system to be meaningful, we need to define F-unforgeable signature schemes, i.e. schemes where it is impossible for an adversary to compute a $(F(msg), \sigma)$ pair on his own.

Our first construction uses the Weak Boneh-Boyen (WBB) signature scheme [BB04]. Using a rather strong assumption, we prove that WBB is F-unforgeable and our P-signature construction is secure. This construction is simple. Our second construction uses a better assumption (because it is falsfiable [Nao03]) and is based on the Full Boneh-Boyen signature scheme [BB04]. We had to modify the Boneh-Boyen construction, however, because the GS proof system would not allow the knowledge extraction of the entire signature.

ORGANIZATION. We define P-signatures in Section 2. Section 3 introduces the complexity assumptions. Section 4 explains non-interactive proofs of knowledge, introduces our new notation, and reviews GS proofs. Finally, Sections 5 and 6 contain our constructions.

2 Definition of a Secure P-Signature Scheme

A P-signature scheme [CL02] lets a user (1) obtain a signature on a committed message without revealing the message, and (2) construct a non-interactive *zero-knowledge proof of knowledge* of $(F(msg), \sigma)$ such that $\mathsf{VerifySig}(pk, msg, \sigma) = \mathsf{accept}$. In this section, we give the first formal definition of a non-interactive P-signature scheme. We begin by reviewing digital signatures and introducing the concept of F-unforgeability.

2.1 Digital Signatures

A signature scheme consists of four algorithms: SigSetup, Keygen, Sign, and VerifySig. $\mathsf{SigSetup}(1^k)$ generates the public parameters $params_{Sig}$. $\mathsf{Keygen}(params_{Sig})$ generates a signing key pair (pk, sk). $\mathsf{Sign}(params_{Sig}, sk, msg)$ computes a signature σ on msg. $\mathsf{VerifySig}(params_{Sig}, pk, msg, \sigma)$ outputs accept if σ is a valid signature on msg, reject otherwise.

The standard definition of a secure signature scheme [GMR88] states that no adversary can output (msg, σ), where σ is a signature on msg, without first previously obtaining a signature on msg (see Appendix B for the GMR definition). This is insufficient for our purposes. Our P-Signature constructions prove that we know some value $y = F(msg)$ (for an efficiently computable bijection F) and a signature σ such that $\mathsf{VerifySig}(params_{Sig}, pk, msg, \sigma) = \mathsf{accept}$. However, even if an adversary cannot output (msg, σ) without first obtaining a signature on msg, he might be able to output $(F(msg), \sigma)$. Therefore, we introduce the notion of F-Unforgeability:

Definition 1 (F-Secure Signature Scheme) *We say that a signature scheme is F-secure (against adaptive chosen message attacks) if it is* Correct *and F-*Unforgeable.

Correct. $\mathsf{VerifySig}$ always accepts a signatures obtained using the Sign algorithm.

F-Unforgeable. Let F be an efficiently computable bijection. No adversary should be able to output the pair $(F(msg), \sigma)$ unless he has previously obtained a signature on msg. Formally, for every PPTM adversary $\mathcal{A}$, there exists a negligible function ν such that

$$\begin{aligned}\Pr[&params_{Sig} \leftarrow \mathsf{SigSetup}(1^k); (pk, sk) \leftarrow \mathsf{Keygen}(params_{Sig});\\ &(Q_{\mathsf{Sign}}, y, \sigma) \leftarrow \mathcal{A}(params_{Sig}, pk)^{\mathcal{O}_{\mathsf{Sign}}(params_{Sig}, sk, \cdot)} :\\ &\mathsf{VerifySig}(params_{Sig}, pk, F^{-1}(y), \sigma) = 1 \wedge y \notin F(Q_{\mathsf{Sign}})] < \nu(k).\end{aligned}$$

$\mathcal{O}_{\mathsf{Sign}}(params_{Sig}, sk, msg)$ records all msg queries on Q_{Sign} and returns $\mathsf{Sign}(params_{Sig}, sk, msg)$. $F(Q_{\mathsf{Sign}})$ evaluates F on all values on Q_{Sign}.

Lemma 1 *An F-unforgeable signature scheme is secure in the standard [GMR88] sense.*

2.2 Commitment Schemes

Recall the standard definition of a non-interactive commitment scheme. It consists of the algorithms $\mathsf{ComSetup}$, Commit. $\mathsf{ComSetup}(1^k)$ outputs the public parameters of the commitment scheme $params_{Com}$. $\mathsf{Commit}(params_{Com}, x, opening)$ is a deterministic function that outputs $comm$, a commitment to x using auxiliary information $opening$. We need commitment schemes that are *perfectly binding* and *computationally hiding*:

Perfectly Binding. For every bitstring $comm$, there exists at most one value x such that there exists opening information $opening$ so that $comm = \mathsf{Commit}(params, x, opening)$. We also require that it be easy to identify the bitstrings $comm$ for which there exists such an x.

Strongly Computationally Hiding. There exists an alternate setup function $\mathsf{HidingSetup}(1^k)$ that outputs parameters (that are computationally indistinguishable from the parameters output by $\mathsf{ComSetup}(1^k)$) so that the commitments become information-theoretically hiding.

2.3 Non-Interactive P-Signatures

A non-interactive P-signature scheme extends a signature scheme (Setup, Keygen, Sign, $\mathsf{VerifySig}$) and a non-interactive commitment scheme (Setup, Commit). It consists of the algorithms (Setup, Keygen, Sign, $\mathsf{VerifySig}$, Commit, $\mathsf{ObtainSig}$, $\mathsf{IssueSig}$, Prove, $\mathsf{VerifyProof}$, $\mathsf{EqCommProve}$, $\mathsf{VerEqComm}$).

$\mathsf{Setup}(1^k)$**.** Outputs public parameters $params$. These parameters include parameters for the signature scheme and the commitment scheme.

$\mathsf{ObtainSig}(params, pk, msg, comm, opening) \leftrightarrow \mathsf{IssueSig}(params, sk, comm)$**.** These two interactive algorithms execute a signature issuing protocol between a user and the issuer. The user takes as input $(params, pk, msg, comm, opening)$ such that the value $comm = \mathsf{Commit}(params, msg, opening)$ and gets a signature σ as output. The issuer gets $(params, sk, comm)$ as input and gets nothing as output.

$\mathsf{Prove}(params, pk, msg, \sigma)$**.** Outputs the values $(comm, \pi, opening)$, such that we have $comm = \mathsf{Commit}(params, msg, opening)$ and π is a proof of knowledge of a signature σ on msg.

$\mathsf{VerifyProof}(params, pk, comm, \pi)$**.** Takes as input a commitment to a message msg and a proof π that the message has been signed by owner of public key pk. Outputs accept if π is a valid proof of knowledge of $F(msg)$ and a signature on msg, and outputs reject otherwise.

$\mathsf{EqCommProve}(params, msg, opening, opening')$ Takes as input a message and two commitment opening values. It outputs a proof π that the commitment $comm = \mathsf{Commit}(msg, opening)$ is a commitment to the same value as $comm' = \mathsf{Commit}(msg, opening')$.

$\mathsf{VerEqComm}(params, comm, comm', \pi)$ takes as input two commitments and a proof and accepts if π is a correct proof that $comm$, $comm'$ are commitments to the same value.

Definition 2 (Secure P-Signature Scheme) *Let F be a efficiently computable bijection (possibly parameterized by public parameters). A P-signature scheme is secure if* ($\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Sign}, \mathsf{VerifySig}$) *form an F-unforgeable signature scheme, if* ($\mathsf{Setup}, \mathsf{Commit}$) *is a Perfectly Binding, Strongly Computationally Hiding commitment scheme, if* ($\mathsf{Setup}, \mathsf{EqCommProve}, \mathsf{VerEqComm}$) *is a non-interactive proof system, and if the* Signer Privacy*,* User privacy*,* Correctness*,* Unforgeability*, and* Zero-Knowledge *properties hold:*

Correctness. An honest user who obtains a P-signature from an honest issuer will be able to prove to an honest verifier that he has a valid signature.

Signer privacy. No PPTM adversary can tell if it is running $\mathsf{IssueSig}$ with an honest issuer or with a simulator who merely has access to a signing oracle. Formally, there exists a simulator $\mathsf{SimIssue}$ such that for all PPTM adversaries $(\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function ν so that:

$$\begin{aligned}
\big|\Pr[&params \leftarrow \mathsf{Setup}(1^k); (sk, pk) \leftarrow \mathsf{Keygen}(params);\\
&(msg, opening, state) \leftarrow \mathcal{A}_1(params, sk);\\
&comm \leftarrow \mathsf{Commit}(params, msg, opening);\\
&b \leftarrow \mathcal{A}_2(state) \leftrightarrow \mathsf{IssueSig}(params, sk, comm) : b = 1]\\
-\Pr[&params \leftarrow \mathsf{Setup}(1^k); (sk, pk) \leftarrow \mathsf{Keygen}(params);\\
&(msg, opening, state) \leftarrow \mathcal{A}_1(params, sk);\\
&comm \leftarrow \mathsf{Commit}(params, msg, opening);\\
&\sigma \leftarrow \mathsf{Sign}(params, sk, msg);\\
&b \leftarrow \mathcal{A}_2(state) \leftrightarrow \mathsf{SimIssue}(params, comm, \sigma) : b = 1]\big| < \nu(k)
\end{aligned}$$

Note that we ensure that $\mathsf{IssueSig}$ and $\mathsf{SimIssue}$ gets an honest commitment to whatever msg, $opening$ the adversary chooses. Since the goal of signer privacy is to prevent the adversary from learning anything except a signature on the opening of the commitment, this is sufficient for our purposes. Note that our SimIssue will be allowed to rewind $\mathcal{A}$. Also, we have defined Signer Privacy in terms of a single interaction between the adversary and the issuer. A simple hybrid argument can be used to show that this definition implies privacy over many sequential instances of the issue protocol.

User privacy. No PPTM adversary $(\mathcal{A}_1, \mathcal{A}_2)$ can tell if it is running $\mathsf{ObtainSig}$ with an honest user or with a simulator. Formally, there exists a simulator $\mathsf{Sim} = \mathsf{SimObtain}$ such that for all PPTM adversaries $\mathcal{A}_1, \mathcal{A}_2$, there exists a negligible function ν so that:

$$\begin{aligned}
\big|\Pr[&params \leftarrow \mathsf{Setup}(1^k); (pk, msg, opening, state) \leftarrow \mathcal{A}_1(params);\\
&comm = \mathsf{Commit}(params, msg, opening);\\
&b \leftarrow \mathcal{A}_2(state) \leftrightarrow \mathsf{ObtainSig}(params, pk, msg, comm, opening) : b = 1]\\
-\Pr[&(params, sim) \leftarrow \mathsf{Setup}(1^k); (pk, msg, opening, state) \leftarrow \mathcal{A}_1(params);\\
&comm = \mathsf{Commit}(params, msg, opening);\\
&b \leftarrow \mathcal{A}_2(state) \leftrightarrow \mathsf{SimObtain}(params, pk, comm) : b = 1]\big| < \nu(k)
\end{aligned}$$

As in Signer Privacy, we ensure that $\mathsf{SimObtain}$ gets an honest commitment. Here again $\mathsf{SimObtain}$ is allowed to rewind the adversary.

Note that we require that only the user's input msg is hidden from the issuer, but not necessarily the user's output σ. The reason that this is sufficient is that in actual applications (for example, in anonymous credentials), a user would never show σ in the clear; instead, he would just prove that he knows σ.

An alternative, stronger way to define signer privacy and user privacy together, would be to require that the pair of algorithms $\mathsf{ObtainSig}$ and $\mathsf{IssueSig}$ carry out a secure two-party computation. This alternative definition would ensure that σ is hidden from the issuer as well. The alternative definition turns out to be much harder to satisfy with an efficient construction. Therefore, we preferred to give a special definition.

Unforgeability. Informally, we require that no PPTM adversary can create a proof for any message msg for which he has not previously obtained a signature or a non-interactive proof.

We say that a P-signature scheme is unforgeable if there exists an extractor (ExtractSetup, Extract) and a bijection F such that (1) the output of ExtractSetup(1^k) is indistinguishable from the output of Setup(1^k), and (2) no PPTM adversary can output a proof π that VerifyProof accepts, but from which we extract $F(msg), \sigma$ such that either (a) σ is not valid signature on msg, or (b) $comm$ is not a commitment to msg or (c) the adversary has never previously queried the signing oracle on msg. Formally, for all PPTM adversaries $\mathcal{A}$, there exists a negligible function ν such that:

$$\begin{aligned}
&\Pr[params_0 \leftarrow \mathsf{Setup}(1^k); (params_1, td) \leftarrow \mathsf{ExtractSetup}(1^k) : b \leftarrow \{0,1\} :\\
&\quad \mathcal{A}(params_b) = b] < 1/2 + \nu(k), \text{ and}\\
&\Pr[(params, td) \leftarrow \mathsf{ExtractSetup}(1^k); (pk, sk) \leftarrow \mathsf{Keygen}(params);\\
&\quad (Q_{\mathsf{Sign}}, comm, \pi) \leftarrow \mathcal{A}(params, pk)^{\mathcal{O}_{\mathsf{Sign}}(params, sk, \cdot)};\\
&\quad (y, \sigma) \leftarrow \mathsf{Extract}(params, td, \pi, comm) :\\
&\quad \mathsf{VerifyProof}(params, pk, comm, \pi) = \mathsf{accept}\\
&\quad \wedge (\mathsf{VerifySig}(params, pk, F^{-1}(y), \sigma) = \mathsf{reject}\\
&\qquad \vee (\forall opening, comm \neq \mathsf{Commit}(params, F^{-1}(y), opening))\\
&\qquad \vee (\mathsf{VerifySig}(params, pk, F^{-1}(y), \sigma) = \mathsf{accept} \wedge y \notin F(Q_{\mathsf{Sign}})))] < \nu(k).
\end{aligned}$$

Oracle $\mathcal{O}_{\mathsf{Sign}}(params, sk, msg)$ returns a signature σ on msg. The oracle runs the function Sign$(params, sk, msg)$ and returns the result to the adversary. It records the queried message on query tape Q_{Sign}. By $F(Q_{\mathsf{Sign}})$ we mean F applied to every message in Q_{Sign}.

Zero-knowledge. There exists a simulator Sim $=$ (SimSetup, SimProve, SimEqCommProve), such that for all PPTM adversaries $\mathcal{A}_1, \mathcal{A}_2$, there exists a negligible function ν such that under parameters output by SimSetup, Commit is perfectly hiding and (1) the parameters output by SimSetup are indistinguishable from those output by Setup, but SimSetup also outputs a special auxiliary string sim; (2) when $params$ are generated by SimSetup, the output of SimProve$(params, sim, pk)$ is indistinguishable from that of Prove$(params, pk, msg, \sigma)$ for all (pk, msg, σ) where $\sigma \in \sigma_{pk}(msg)$; and (3) when $params$ are generated by SimSetup, the output of SimEqCommProve$(params, sim, comm, comm')$ is indistinguishable from that of EqCommProve$(params, msg, opening, opening')$ for all $(msg, opening, opening')$ where $comm = \mathsf{Commit}(params, msg, opening)$ and $comm' = \mathsf{Commit}(params, msg, opening')$.

In GMR notation, this is formally defined as follows:

$$\begin{aligned}
&|\Pr[params \leftarrow \mathsf{Setup}(1^k); b \leftarrow \mathcal{A}(params) : b = 1]\\
&\quad - \Pr[(params, sim) \leftarrow \mathsf{SimSetup}(1^k); b \leftarrow \mathcal{A}(params) : b = 1]| < \nu(k), \text{ and}\\
&|\Pr[(params, sim) \leftarrow \mathsf{SimSetup}(1^k); (pk, msg, \sigma, state) \leftarrow \mathcal{A}_1(params, sim);\\
&\quad (comm, \pi, opening) \leftarrow \mathsf{Prove}(params, pk, msg, \sigma); b \leftarrow \mathcal{A}_2(state, comm, \pi) : b = 1]\\
&\quad - \Pr[(params, sim) \leftarrow \mathsf{SimSetup}(1^k); (pk, msg, \sigma, state) \leftarrow \mathcal{A}_1(params, sim);\\
&\quad (comm, \pi) \leftarrow \mathsf{SimProve}(params, sim, pk); b \leftarrow \mathcal{A}_2(state, comm, \pi) : b = 1]| < \nu(k), \text{ and}\\
&|\Pr[(params, sim) \leftarrow \mathsf{SimSetup}(1^k); (msg, opening, opening') \leftarrow \mathcal{A}_1(params, sim);\\
&\quad \pi \leftarrow \mathsf{EqCommProve}(params, msg, opening, opening'); b \leftarrow \mathcal{A}_2(state, \pi) : b = 1]\\
&\quad - \Pr[(params, sim) \leftarrow \mathsf{SimSetup}(1^k); (msg, opening, opening') \leftarrow \mathcal{A}_1(params, sim);\\
&\quad \pi \leftarrow \mathsf{SimEqCommProve}(params, sim, \mathsf{Commit}(params, msg, opening), \mathsf{Commit}(params, msg, opening'));\\
&\quad b \leftarrow \mathcal{A}_2(state, \pi) : b = 1]| < \nu(k).
\end{aligned}$$

3 Preliminaries

We say that a function $\nu : \mathbb{Z} \rightarrow \mathbb{R}$ is negligible if for all integers c there exists an integer K such that $\forall k > K$, $|\nu(k)| < 1/k^c$. We use the standard GMR [GMR88] notation to describe probability spaces.

Let G_1, G_2 and G_T be groups. A function $e : G_1 \times G_2 \rightarrow G_T$ is called *a cryptographic bilinear map* if it has the following properties: *Bilinear.* $\forall a \in G_1, \forall b \in G_2, \forall x, y \in \mathbb{Z}$ the following equation holds: $e(a^x, b^y) = e(a, b)^{xy}$. *Non-Degenerate.* If a and b are generators of their respective groups, then $e(a, b)$ generates the group G_T. *One-Way.* Let $\mathsf{BilinearSetup}(1^k)$ be an algorithm that generates the groups G_1, G_2 and G_T, together with algorithms for sampling from these groups, and the algorithm for computing the function e.

The function $\mathsf{BilinearSetup}(1^k)$ outputs $params_{BM} = (p, G_1, G_2, G_T, e, g, h)$, where p is a prime (of length k), G_1, G_2, G_T are groups of order p, g is a generator of G_1, h is a generator of G_2, and $e : G_1 \times G_2 \rightarrow G_T$ is a bilinear map.

We informally state the cryptographic assumptions we use. See Appendix D for formal definitions and discussion. We introduce two new assumptions we call IHSDH and TDH and review the HSDH assumption introduced by Boyen and Waters [BW07b].

Definition 3 (Interactive Hidden SDH assumption (IHSDH).) *No PPTM adversary can compute a tuple* $(g^{1/(x+c)}, h^c, u^c)$ *given* (g, g^x, h, h^x, u) *and permission to make* q *queries to oracle* $\mathcal{O}_x(c)$ *that returns* $g^{1/(x+c)}$*. The* c *used by the adversary must be different from the values it used to query* $\mathcal{O}_x(\cdot)$*.*

Definition 4 (Triple DH (TDH)) *On input* $g, g^x, g^y, h, h^x, \{c_i, g^{1/(x+c_i)}\}_{i=1...q}$*, it is computationally infeasible to output a tuple* $(h^{\mu x}, g^{\mu y}, g^{\mu xy})$ *for* $\mu \neq 0$*.*

Definition 5 (Hidden SDH [BW07b]) *On input* $g, g^x, u \in G_1$*,* $h, h^x \in G_2$ *and* $\{g^{1/(x+c_\ell)}, h^{c_\ell}, u^{c_\ell}\}_{\ell=1...q}$*, it is computationally infeasible to output a new tuple* $(g^{1/(x+c)}, h^c, u^c)$*.*

Groth-Sahai proofs use either the DLIN or XDH assumption.

4 Non-Interactive Proofs of Knowledge

Our P-signature constructions use the Groth and Sahai [GS07] non-interactive proof of knowledge (NIPK) system. De Santis et al. [SCP00] give the standard definition of NIPK systems. Their definition does not fully cover the Groth and Sahai proof system. In this section, we review the standard notion of NIPK. Then we give a useful generalization, which we call an f-extractable NIPK, where the extractor only extracts a function of the witness. We develop useful notation for expressing f-extractable NIPK systems, and explain how this notation applies to the Groth-Sahai construction. We then review Groth-Sahai commitments and pairing product equation proofs. Finally, we show how they can be used to prove statments about committed exponents, as this will be necessary later for our constructions.

4.1 Proofs of Knowledge: Notation and Definitions

In this subsection, we review the definition of NIPK, introduce the notion of f-extractability, and develop some useful notation.

We review the De Santis et al. [SCP00] definition of NIPK. Let $L = \{s : \exists x \text{ s.t. } M_L(s, x) = \mathsf{accept}\}$ be a language in NP and M_L a polynomial-time Turing Machine that verifies that x is a valid witness for the statement $s \in L$.[1] A NIPK system consists of three algorithms: (1) $\mathsf{PKSetup}(1^k)$ sets up the common parameters $params_{PK}$; (2) $\mathsf{PKProve}(params_{PK}, s, x)$ computes a proof π of the statement $s \in L$ using witness x; (3) $\mathsf{PKVerify}(params_{PK}, s, \pi)$ verifies correctness of π. The system must be *complete* and *extractable*. Completeness means that for all values of $params_{PK}$ and for all s, x such that $M_L(s, x) = \mathsf{accept}$, a proof π generated by $\mathsf{PKProve}(params_{PK}, s, x)$ must be accepted by $\mathsf{PKVerify}(params_{PK}, s, \pi)$. Extractability means that there exists a polynomial-time extractor $(\mathsf{PKExtractSetup}, \mathsf{PKExtract})$. $\mathsf{PKExtractSetup}(1^k)$ outputs $(td, params_{PK})$ where $params_{PK}$ is distributed

[1] We use x to denote a witness to stay consistent with Groth and Sahai [GS07].

identically to the output of $\mathsf{PKSetup}(1^k)$. For all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}(1^k, params_{PK})$ outputs (s, π) such that $\mathsf{PKVerify}(params_{PK}, s, \pi) = \mathsf{accept}$ and $\mathsf{PKExtract}(td, s, \pi)$ fails to extract a witness x such that $M_L(s, x) = \mathsf{accept}$ is negligible in k. We have *perfect* extractability if this probability is 0.

We first generalize the notion of NIPK for a language L to languages parameterized by $params_{PK}$ – we allow the Turing machine M_L to receive $params_{PK}$ as a separate input.

Next, we generalize extractability to f-extractability. We say that a NIPK system is f-extractable if $\mathsf{PKExtract}$ outputs y, such that there $\exists x : M_L(params_{PK}, s, x) = \mathsf{accept} \wedge y = f(params_{PK}, x)$. If $f(params_{PK}, \cdot)$ is the identity function, we get the usual notion of extractability. We denote an f-extractable proof π obtained by running $\mathsf{PKProve}(params_{PK}, s, x)$ as

$$\pi \leftarrow \mathsf{NIPK}\{params_{PK}, s, f(params_{PK}, x) : M_L(params_{PK}, s, x) = \mathsf{accept}\}.$$

We omit the $params_{PK}$ where they are obvous. In our applications, s is actually a conditional statement about the properties of the witness x. So $M_L(s, x) = \mathsf{accept}$ if $\mathsf{Condition}(x) = \mathsf{accept}$. Thus the statement $\pi \leftarrow \mathsf{NIPK}\{f(x) : \mathsf{Condition}(x)\}$ is well defined. Suppose s includes a list of commitments $c_n = \mathsf{Commit}(x_n, opening_n)$. The witness is $x = (x_1, \ldots, x_N, opening_1, \ldots, opening_N)$, however, we typically can only extract $x_1, \ldots, x_N$. We write

$$\pi \leftarrow \mathsf{NIPK}\{(x_1, \ldots, x_n) : \mathsf{Condition}(x) \wedge \forall \ell : c_\ell = \mathsf{Commit}(params_{Com}, x_\ell, opening_\ell)\}.$$

We introduce shorthand notation for the above expression: $\pi \leftarrow \mathsf{NIPK}\{((c_1 : x_1), \ldots, (c_n : x_n)) : \mathsf{Condition}(x)\}$. For simplicity, we assume the proof π includes s.

4.2 Groth-Sahai Commitments [GS07]

We review the Groth-Sahai [GS07] commitment scheme. We use their scheme to commit to elements of a group G of prime order p. Technically, their constructions commit to elements of certain modules, but we can apply them to certain bilinear groups (see Appendix E). Groth and Sahai also have constructions for composite order groups using the Subgroup Decision assumption; we cannot use them because they do not have certain extraction properties.

$\mathsf{GSComSetup}(p, G, g)$**.** Outputs a common reference string $params_{Com}$.

$\mathsf{GSCommit}(params_{Com}, x, opening)$**.** Takes as input $x \in G$ and some value $opening$ and outputs a commitment $comm$. The extension $\mathsf{GSCommit}(params_{Com}, b, \theta, opening)$ takes as input $\theta \in Z_p$ and a base $b \in G$ and outputs $(b, comm)$, where $comm = \mathsf{GSCommit}(params_{Com}, b^\theta, opening)$. (Groth and Sahai compute commitments to elements in Z_p slightly differently; Our method allows us to prove equality of exponents committed using different bases in Section 4.4.)

$\mathsf{VerifyOpening}(params_{Com}, comm, x, opening)$**.** Takes $x \in G$ and $opening$ as input and outputs accept if $comm$ is a commitment to x. To verify that $(b, comm)$ is a commitment to exponent θ check $\mathsf{VerifyOpening}(params_{Com}, comm, b^\theta, opening)$.

For brevity, we write $\mathsf{GSCommit}(x)$ to indicate committing to $x \in G$ when the parameters are obvious and the value of $opening$ is chosen appropriately at random. Similarly, $\mathsf{GSCommit}(b, \theta)$ indicates committing to θ using $b \in G$ as the base.

GS commitments are *perfectly binding*, *strongly computationally hiding*, and *extractable*.

Groth and Sahai [GS07] show how to instantiate commitments that meet the above requirements using either the SXDH or DLIN assumptions. Commitments based on SXDH consist of 2 elements in G, while those based on DLIN setting require 3 elements in G. Note that in the Groth-Sahai proof system below, $G = G_1$ or $G = G_2$ for SXDH and $G = G_1 = G_2$ for DLIN.

4.3 Groth-Sahai Pairing Product Equation Proofs [GS07]

Groth and Sahai [GS07] construct an f-extractable NIPK system that lets us prove statements in the context of groups with bilinear maps. We first give intuition about their result, then provide a formal definition.

$\mathsf{GSSetup}(1^k)$ outputs $(p, G_1, G_2, G_T, e, g, h)$, where G_1, G_2, G_T are groups of prime order p, with g a generator of G_1, h a generator of G_2, and $e : G_1 \times G_2 \to G_T$ a cryptographic bilinear map. $\mathsf{GSSetup}(1^k)$ also outputs $params_1$ and $params_2$ for constructing GS commitments in G_1 and G_2, respectively. (If the pairing is symmetric, $G_1 = G_2$ and $params_1 = params_2$.) The statement s to be proven consists of the following list of values: $\{a_q\}_{q=1\ldots Q} \in G_1$, $\{b_q\}_{q=1\ldots Q} \in G_2$, $t \in G_T$, and $\{\alpha_{q,m}\}_{m=1\ldots M, q=1\ldots Q}$, $\{\beta_{q,n}\}_{n=1\ldots N, q=1\ldots Q} \in Z_p$, as well as a list of commitments $\{c_m\}_{m=1\ldots M}$ to values in G_1 and $\{d_n\}_{n=1\ldots N}$ to values in G_2. Groth and Sahai show how to construct the following proof:

$$NIPK\{((c_1 : x_1), \ldots, (c_M : x_M), (d_1 : y_1), \ldots, (d_N : y_N)) : \\ \prod_{q=1}^{Q} e(a_q \prod_{m=1}^{M} x_m^{\alpha_{q,m}}, b_q \prod_{n=1}^{N} y_n^{\beta_{q,n}}) = t$$

The proof π includes the statement being proven; this includes the commitments $c_1, \ldots, c_M$ and $d_1, \ldots, d_N$. Groth and Sahai provide an efficient extractor that opens these commitments to values $x_1, \ldots, x_M, y_1, \ldots, y_N$ that satisfy the pairing product equation.

Recall the function $\mathsf{GSCommit}(params_1, b, \theta, opening) = (b, \mathsf{GSCommit}(params_1, b^\theta, opening))$. We can replace any of the clauses $(c_m : x_m)$ with the clause $(c_m : b^\theta)$, and add b to the list of values included in the statement s (and therefore in the proof π). The same holds for commitments d_n. Groth-Sahai proofs also allow us to prove that the openings of $(c_1, \ldots, c_n, d_1, \ldots, d_n)$ satisfy several equations *simultaneously*.

We formally define the Groth-Sahai proof system. Let $params_{BM} \leftarrow \mathsf{BilinearSetup}(1^k)$.

$\mathsf{GSSetup}(params_{BM})$. Calls $\mathsf{GSComSetup}$ to generate $params_1$ and $params_2$ that let it construct commitments in G_1 and G_2 respectively. It may also calculates some auxiliary values $params_\pi$. Outputs $params_{GS} = (params_{BM}, params_1, params_2, params_\pi)$.

$\mathsf{GSProve}(params_{GS}, s, (\{x_m\}_{1\ldots M}, \{y_n\}_{1\ldots N}, openings))$. Takes as input the parameters, the statement $s = \{(c_1, \ldots, c_M, d_1, \ldots, d_N), equations\}$ to be proven, (the statement s includes the commitments and the parameters of the pairing product equations), the witness consisting of the values $\{x_m\}_{1\ldots M}, \{y_n\}_{1\ldots N}$ and opening information $openings$. Outputs a proof π.

$\mathsf{GSVerify}(params_{GS}, \pi)$. Returns accept if π is valid, reject otherwise. (Note that it does not take the statement s as input because we have assumed that the statement is always included in the proof π.)

$\mathsf{GSExtractSetup}(params_{BM})$. Outputs $params_{GS}$ and auxiliary information (td_1, td_2). $params_{GS}$ are distributed identically to the output of $\mathsf{GSSetup}(params_{BM})$. that allows an extractor to discover the contents of all commitments.

$\mathsf{GSExtract}(params_{GS}, td_1, td_2, \pi)$ outputs $x_1, \ldots, x_M$ and $y_1, \ldots, y_N$ that satisfy the *equations* and that correspond to the commitmets (note that the commitments and the equations are included with the proof π).

Groth-Sahai proofs satisfy *correctness*, *extractability*, and *strong witness indistinguishability*. We explain what these requirements entail in a manner compatible with our notation.

Correctness. An honest verifier always accepts a proof generated by an honest prover.

Extractability. If an honest verifier outputs accept, then the statement is true. This means that, given td_1, td_2 corresponding to $params_{GS}$, the $\mathsf{GSExtract}$ extracts values from the commitments that will satisfy the pairing product equations with probability 1.

Strong Witness Indistinguishability. There exists a simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimProve})$ with the following two properties: (1) $\mathsf{SimSetup}(params_{BM})$ outputs $params_{GS}'$ such that they are computationally indistinguishable from the output of $\mathsf{GSSetup}(params_{BM})$. Let $params_1' \in params_{GS}'$ be the parameters for the commitment scheme in G_1. Using $params_1'$, commitments are perfectly hiding – this means that for all commitments $comm$, $\forall x \in G, \exists opening : \mathsf{VerifyOpening}(params_1', comm, x, opening) = \mathsf{accept}$. (2) Using the $params_{GS}'$ generated by the challenger, GS proofs become perfectly witness indistinguishable. Suppose an unbounded adversary $\mathcal{A}$ generates a statement s consisting of the pairing product equations and a set of commitments $(c_1, \ldots, c_M, d_1, \ldots, d_N)$. Next the adversary opens the commitments in two different ways $W_0 =$

$(x_1^{(0)}, \ldots, x_M^{(0)}, y_1^{(0)}, \ldots, y_N^{(0)}, openings_0)$ and $W_1 = (x_1^{(1)}, \ldots, x_M^{(1)}, y_1^{(1)}, \ldots, y_N^{(1)}, openings_1)$ (with the requirement that these witnesses must both satisfy s). The values $openings_b$ show how to open the commitments to $\{x_m^{(b)}, y_n^{(b)}\}$. (The adversary can do this because it is unbounded.) The challenger gets the statement s and the two witnesses W_0 and W_1. He chooses a bit $b \leftarrow \{0, 1\}$ and computes $\pi = \mathsf{GSProve}(params_{GS}', s, W_b)$. Strong witness indistinguishability means that π is distributed independently of b.

Composable Zero-Knowledge. In some contexts, GS pairing product equation proofs are composable zero-knowledge. See Appendix G.1 for the definition.

Efficiency. See Appendix J for an analysis of the efficiency of GS pairing product equation proofs.

4.4 Proofs about Committed Exponents

We use Groth-Sahai pairing product equations to prove equality of committed exponents.

Equality of Committed Exponents in Different Groups. We want to prove the statement $\mathsf{NIPK}\{((c : g^\alpha), (d : h^\beta)) : \alpha = \beta\}$. We perform a Groth-Sahai pairing product equation proof $\mathsf{NIPK}\{((c : x), (d : y)) : e(x, h)e(1/g, y) = 1\}$. Security is straightforward due to the f-extractability property of the GS proof system.

Equality of Committed Exponents in the Same Group. We want to prove the statement $\mathsf{NIPK}\{((c_1 : g^\alpha), (c_2 : u^\beta)) : \alpha = \beta\}$, where $g, u \in G_1$. This is equivalent to proving $\mathsf{NIPK}\{((c_1 : g^\alpha), (c_2 : u^\beta), (d : h^\gamma) : \alpha = \gamma \wedge \beta = \gamma\}$.

Zero-Knowledge Proof of Equality of Committed Exponents. We want to prove the statement $NIZKPK\{((c_1 : g^\alpha), (c_2 : g^\beta) : \alpha = \beta\}$ in zero-knowledge. We perform the Groth-Sahai pairing product equation proof $\mathsf{NIPK}\{((c_1 : g^\alpha), (c_2 : g^\beta), (d : h^\theta) : e(a/b, h^\theta) = 1 \wedge e(g, h^\theta)e(1/g, h) = 1\}$. Proof of equality of committed exponents in group G_2 is done analogously. See Appendix G.2 for details.

Remark. We cannot use Groth-Sahai general arithmetic gates [GS07] to construct the above proofs because they only work when the opening of commitments have the same base.

5 First Construction of P-Signature Scheme

Our first construction of a P-signature scheme uses the Weak Boneh-Boyen signature scheme (WBB) signature scheme [BB04] as a building block. The WBB scheme is as follows:

WBB-$\mathsf{SigSetup}(1^k)$ runs $\mathsf{BilinearSetup}(1^k)$ to get the pairing parameters $(p, G_1, G_2, G_T, e, g, h)$. In the sequel, by z we denote $z = e(g, h)$.

WBB-$\mathsf{Keygen}(params_{Sig})$ The secret key is $\alpha \leftarrow Z_p$. $pk = (v, \tilde{v})$, where $v = h^\alpha$, $\tilde{v} = g^\alpha$.[2] The correctness of the public key can be verified by checking that $e(g, v) = e(\tilde{v}, h)$.

WBB-$\mathsf{Sign}(params_{Sig}, sk, msg)$ calculates $\sigma = g^{1/(\alpha+msg)}$, where $sk = \alpha$.

WBB-$\mathsf{VerifySig}(params_{Sig}, pk, msg, \sigma)$ outputs accept if the public key is correctly formed and if $e(\sigma, vh^{msg}) = z$, where $pk = (v, \tilde{v})$. Outputs reject otherwise.

Boneh and Boyen proved that the Weak Boneh-Boyen signature is only weakly secure given SDH, which is insufficient for our purposes. In Appendix H, we show that the weak Boneh-Boyen signature scheme is F-secure given IHSDH (which implies standard [GMR88] security).

Theorem 1 *Let $F(x) = (h^x, u^x)$, where $u \in G_1$ and $h \in G_2$ as given in the statement of the IHSDH assumption. The Weak Boneh-Boyen signature scheme is F-secure given IHSDH.*

[2] The shadow value $\tilde{v}^\alpha$ does not exist in [BB04] and is needed to prove zero-knowledge of our P-signatures in pairing settings in which no efficient isomorphisms exist.

We extend the WBB scheme to obtain our first P-signature (Setup, Keygen, Sign, VerifySig, Commit, ObtainSig, IssueSig, Prove, VerifyProof, EqCommProve, VerEqComm), as follows:

Setup(1^k) First, obtain $params_{BM} = (p, G_1, G_2, G_T, e, g, h) \leftarrow$ BilinearSetup(1^k). Next, obtain $params_{GS} = (params_{BM}, params_1, params_2, params_\pi) \leftarrow$ GSSetup($params_{BM}$). Pick $u \leftarrow G_1$. Let $params = (params_{GS}, u)$. As before, z is defined as $z = e(g, h)$.

Keygen($params$) Run WBB-Keygen($params_{BM}$) and outputs $sk = \alpha$, $pk = (h^\alpha, g^\alpha) = (v, \tilde{v})$.

Sign($params, sk, msg$) Run WBB-Sign($params_{BM}, sk, msg$) to obtain $\sigma = g^{1/(\alpha+msg)}$ where $\alpha = sk$.

VerifySig($params, pk, msg, \sigma$) Run WBB-VerifySig($params_{Sig}, pk, msg, \sigma$).

Commit($params, msg, opening$) To commit to msg, compute C = GSExpCommit($params_2, h, msg, opening$). (Recall that GSExpCommit($params_2, h, msg, opening$) = GSCommit($params_2, h^{msg}, opening$), and $params_2$ is part of $params_{GS}$.)

ObtainSig($params, pk, msg, comm, opening$) $\leftrightarrow$ IssueSig($params, sk, comm$)**.** The user and the issuer run the following protocol:

1. The user chooses $\rho \leftarrow Z_p$.
2. The user and issuer engage in a secure two-party computation protocol [JS07][3], where the user's private input is ($\rho, msg, opening$), and the issuer's private input is $sk = \alpha$.
   The issuer's private output is $x = (\alpha + msg)\rho$ if $comm$ = Commit($params, msg, opening$), and $x = \perp$ otherwise.
3. If $x \neq \perp$, the issuer calculates $\sigma' = g^{1/x}$ and sends σ' to the user.
4. The user computes $\sigma = \sigma'^\rho = g^{1/(\alpha+msg)}$. The user checks that the signature is valid.

Prove($params, pk, msg, \sigma$) Check if pk and σ are valid, and if they are not, output $\perp$. Else, pick appropriate $opening_1, opening_2, opening_3$ and form the following three GS commitments: M_h = GSExpCommit($params_2, h, msg, opening_1$), M_u = GSExpCommit($params_1, u, msg, opening_2$), Σ = GSCommit($params_1, \sigma, opening_3$). Compute the following proof: π = NIPK$\{((M_h : h^\alpha), (M_u : u^\beta), (\Sigma : x)) : \alpha = \beta \wedge e(x, vh^\alpha) = z\}$. Output $(comm, \pi) = (M_h, \pi)$.

VerifyProof($params, pk, comm, \pi$) Outputs accept if the proof π is a valid proof of the statement described above for $M_h = comm$ and for properly formed $pk = (v, \tilde{v})$.

EqCommProve($params, msg, opening, opening'$) Let commitment $comm$ = Commit($params, msg, opening$) = GSCommit($params_2, h^{msg}, opening$) and $comm'$ = Commit($params, msg, opening'$) = GSCommit($params_2$, $h^{msg}, opening'$). Use the GS proof system as described in Section 4.4 to compute $\pi \leftarrow NIZKPK\{((comm : h^\alpha), (comm' : h^\beta) : \alpha = \beta\}$.

VerEqComm($params, comm, comm', \pi$) Verify the proof π using the GS proof system as described in Section 4.4.

Theorem 2 (Efficiency) *Using SXDH, each P-signature proof for the weak Boneh-Boyen signature scheme consists of* 12 *elements in* G_1 *and* 10 *elements in* G_2*. The prover performs* 22 *multi-exponentiations and the verifier* 44 *pairings. Using DLIN, each P-signature proof consists of* 27 *elements in* $G_1 = G_2$*. The prover performs* 27 *multi-exponentiations and the verifier* 54 *pairings. See Appendix J for details.*

Theorem 3 (Security) *Our first P-signature construction is secure given IHSDH and the security of the GS commitments and proofs.*

[3] Jarecki and Shmatikov give a protocol for secure two-party computation on committed inputs; their construction can be adapted here. In general using secure two-party computation is expensive, but here we only need to compute a relatively small and simple circuit on the inputs.

Proof. Correctness follows from correctness of GS proofs.

Signer Privacy. We must construct the SimIssue algorithm that is given as input $params$, a commitment $comm$ and a signature σ and must simulate the adversary's view. SimIssue will invoke the simulator for the two-party computation protocol. Recall that in two-party computation, the simulator can first extract the input of the adversary: in this case, some $(\rho, msg, opening)$. Then SimIssue checks that $comm = \mathsf{Commit}(params, msg, opening)$; if it isn't, it terminates. Otherwise, it sends to the adversary the value $\sigma' = \sigma^{1/\rho}$. Suppose the adversary can determine that it is talking with a simulator. Then it must be the case that the adversary's input to the protocol was incorrect which breaks the security properties of the two-party computation.

User privacy. The simulator will invoke the simulator for the two-party computation protocol. Recall that in two-party computation, the simulator can first extract the input of the adversary (in this case, some α', not necessarily the valid secret key). Then the simulator is given the target output of the computation (in this case, the value x which is just a random value that the simulator can pick itself), and proceeds to interact with the adversary such that if the adversary completes the protocol, its output is x. Suppose the adversary can determine that it is talking with a simulator. Then it breaks the security of the two-party computation protocol.

Zero knowledge. Consider the following algorithms. SimSetup runs $\mathsf{BilinearSetup}(1^k)$ to get $params_{BM} = (p, G_1, G_2, G_T, e, g, h)$ and $\mathsf{GSSimSetup}(params_{BM})$ to get $params_{GS}, sim_{GS}$. It then picks $t \leftarrow Z_p$ and sets up $u = g^t$. The final parameters are $params = (params_{GS}, u, z = e(g, h))$ and $sim = (t, sim_{GS})$. Note that the distribution of $params$ is indistinguishable from the distribution output by Setup. Also note that using these parameters, the commitments generated by GSCommit are perfectly hiding.

SimProve receives $params$, sim, and public key $(v, \tilde{v})$ and can use trapdoor $sim = t$ to create a random P-signature forgery as follows. Pick $s \leftarrow Z_p$ and compute $\sigma = g^{1/s}$. We implicitly set $msg = s - \alpha$. Note that the simulator does not know msg and α. However, he can compute $h^{msg} = h^s/v$ and $u^{msg} = (g^s/\tilde{v})^t$. Now he can use σ, h^{msg}, and u^{msg} to create commitments. The proof π is computed in the same way as in the real Prove protocol using σ, h^{msg}, and u^{msg} and the opening information of the commitments as witnesses. By the witness indistinguishability of the GS proof system, a proof using the faked witnesses is indistinguishable from a proof using a real witness, thus SimProve is indistinguishable from Prove.

Finally, we need to show that we can simulate proofs of EqCommProve given the trapdoor sim_{GS}. This follows from composable zero knoweldge of EqCommProve. See Appendix G.

Unforgeability. Consider the following algorithms: $\mathsf{ExtractSetup}(1^k)$ outputs the usual $params$, except that it invokes GSExtractSetup to get alternative $params_{GS}$ and the trapdoor $td = (td_1, td_2)$ for extracting from GS commitments in G_1 and G_2. The parameters generated by GSSetup are indistinguishable from those generated by GSExtractSetup, so we know that the parameters generated by ExtractSetup will be indistinguishable from those genrated by Setup.

$\mathsf{Extract}(params, td, comm, \pi)$ extracts the values from commitment $comm$ and the commitments M_h, M_u contained in the proof π using the GS commitment extractor. If VerifyProof accepts then $comm = M_h$. Let $F(msg) = (h^{msg}, u^{msg})$.

Now suppose we have an adversary that can break the unforgeability of our P-signature scheme for this extractor and this bijection. We create a reduction to break the IHSDH assumption. The reduction gets $(p, G_1, G_2, G_T, e, g, \tilde{X}, h, X, u)$, where $X = h^x, \tilde{X} = g^x$ for some unknown x. The reduction runs $\mathsf{GSExtractSetup}(p, G_1, G_2, G_T, e, g, h)$ to get $params_{GS}$ and td. It otherwise creates $params$ in the same way as Setup (and ExtractSetup). Note that td lets it open all commitments. The reduction gives $(params, pk = (X, \tilde{X})$ to the adversary. Whenever the adversary queries $\mathcal{O}_{\mathsf{Sign}}$ on msg, the reduction returns $\sigma \leftarrow \mathcal{O}_x(msg)$ and stores msg in Q_{Sign}.

Eventually, the adversary outputs a proof π. Since π is f-extractable and perfectly sound, $\mathsf{Extract}(params, td, comm, \pi)$ will return $a = h^m$, $b = u^m$, and $\sigma = g^{1/(x+m)}$. Thus we have a valid IHSDH tuple and $m = F^{-1}(a, b)$ will always fulfill VerifySig. We also know that since VerifyProof accepts, $comm = M_h = \mathsf{Commit}(params, m, opening)$ for some $opening$. Thus, since this is a forgery, it must be the case that $(a, b) = F(m) \notin F(Q_{\mathsf{Sign}})$. This means that we never queried $\mathcal{O}_x$ on m and the reduction has generated a fresh IHSDH tuple. □

6 Second Construction of P-Signature Scheme

In this section, we present a new signature scheme and then build a P-signature scheme from it. The new signature scheme is inspired by the full Boneh-Boyen signature scheme, and is as follows:

New-$\mathsf{SigSetup}(1^k)$ Same as WBB-$\mathsf{SigSetup}(1^k)$.

New-$\mathsf{Keygen}(params)$ picks a random $\alpha, \beta \leftarrow Z_p$. The signer calculates $v = h^\alpha$, $w = h^\beta$, $\tilde{v} = g^\alpha$, $\tilde{w} = g^\beta$. The secret-key is $sk = (\alpha, \beta)$. The public-key is $pk = (v, w, \tilde{v}, \tilde{w})$. The public key can be verified by checking that $e(g, v) = e(\tilde{v}, h)$ and $e(g, w) = e(\tilde{w}, h)$.

New-$\mathsf{Sign}(params, (\alpha, \beta), msg)$ chooses $r \leftarrow Z_p - \{\frac{\alpha - msg}{\beta}\}$ and calculates $C_1 = g^{1/(\alpha + msg + \beta r)}$, $C_2 = w^r$, $C_3 = u^r$. The signature is (C_1, C_2, C_3).

New-$\mathsf{VerifySig}(params, (v, w, \tilde{v}, \tilde{w}), msg, (C_1, C_2, C_3))$ outputs accept if $e(C_1, vh^{msg}C_2) = z$, $e(u, C_2) = e(C_3, w)$, and if the public key is correctly formed, i.e., $e(g, v) = e(\tilde{v}, h)$, and $e(g, w) = e(\tilde{w}, h)$.[4]

Theorem 4 *Let $F(x) = (h^x, u^x)$, where $u \in G_1$ and $h \in G_2$ as in the HSDH and TDH assumptions. Our new signature scheme is F-secure given HSDH and TDH.*

Proof. See Appendix I for proof. □

We extend the above signature scheme to obtain our second P-signature (Setup, Keygen, Sign, $\mathsf{VerifySig}$, Commit, $\mathsf{ObtainSig}$, $\mathsf{IssueSig}$, Prove, $\mathsf{VerifyProof}$, $\mathsf{EqCommProve}$, $\mathsf{VerEqComm}$). The algorithms Setup, Commit, $\mathsf{EqCommProve}$ and $\mathsf{VerEqComm}$ are the same as in the first construction in Section 5. The rest are as follows:

$\mathsf{Keygen}(params)$ Runs the New-$\mathsf{Keygen}(params_{BM})$ and outputs $sk = (\alpha, \beta)$, $pk = (h^\alpha, h^\beta, g^\alpha, g^\beta) = (v, w, \tilde{v}, \tilde{w})$.

$\mathsf{Sign}(params, sk, msg)$ Run New-$\mathsf{Sign}(params_{BM}, sk, msg)$ to obtain $\sigma = (C_1, C_2, C_3)$ where $C_1 = g^{1/(\alpha + msg + \beta r)}$, $C_2 = w^r$, $C_3 = u^r$, and $sk = (\alpha, \beta)$

$\mathsf{VerifySig}(params, pk, msg, \sigma)$ Run New-$\mathsf{VerifySig}(params_{Sig}, pk, msg, \sigma)$.

$\mathsf{ObtainSig}(params, pk, msg, comm, opening) \leftrightarrow \mathsf{IssueSig}(params, sk, comm)$**.** The user and the issuer run the following protocol:

1. The user chooses $\rho_1, \rho_2 \leftarrow Z_p$.
2. The issuer chooses $r' \leftarrow Z_p$.
3. The user and the issuer run a secure two-party computation protocol where the user's private inputs are $(\rho_1, \rho_2, msg, opening)$, and the issuer's private inputs are $sk = (\alpha, \beta)$ and r'.
   The issuer's private output is $x = (\alpha + msg + \beta\rho_1 r')\rho_2$ if $comm = \mathsf{Commit}(params, msg, opening)$, and $x = \perp$ otherwise.
4. If $x \neq \perp$, the issuer calculates $C_1' = g^{1/x}$, $C_2' = w^{r'}$ and $C_3' = u^{r'}$, and sends (C_1', C_2', C_3') to the user.
5. The user computes $C_1 = C_1'^{\rho_2}$, $C_2 = C_2'^{\rho_1}$, and $C_3 = C_3'^{\rho_1}$ and then verifies that the signature (C_1, C_2, C_3) is valid.

$\mathsf{Prove}(params, pk, msg, \sigma)$ Check if pk and σ are valid, and if they are not, output $\perp$. Then the user computes commitments $\Sigma = \mathsf{GSCommit}(params_1, C_1, opening_1)$, $R_w = \mathsf{GSCommit}(params_1, C_2, opening_2)$, $R_u = \mathsf{GSCommit}(params_1, C_3, opening_3)$, $M_h = \mathsf{GSExpCommit}(params_2, h, msg, opening_4) = \mathsf{GSCommit}(params_2, h^{msg}, opening_4)$ and $M_u = \mathsf{GSExpCommit}(params_1, u, msg, opening_5) = \mathsf{GSCommit}(params_1, u^{msg}, opening_5)$.

The user outputs the commitment $comm = M_h$ and the proof

$$\pi = \mathsf{NIPK}\{((\Sigma : C_1), (R_w : C_2), (R_u : C_3)(M_h : h^\alpha), (M_u : u^\beta)) : e(C_1, vh^\alpha C_2) = z \wedge e(u, C_2) = e(C_3, w) \wedge \alpha = \beta\}.$$

[4]The latter is needed only once per public key, and is meaningless in a symmetric pairing setting.

$\mathsf{VerifyProof}(params, pk, comm, \pi)$ Outputs accept if the proof π is a valid proof of the statement described above for $M_h = comm$ and for properly formed pk.

$\mathsf{EqCommProve}, \mathsf{VerEqComm}$ are as in the first P-signature scheme.

Theorem 5 (Efficiency) *Using SXDH GS proofs, each P-signature proof for our new signature scheme consists of 18 elements in G_1 and 16 elements in G_2. The prover performs 34 multi-exponentiation and the verifier 68 pairings. Using DLIN, each P-signature proof consists of 42 elements in $G_1 = G_2$. The prover has to do 42 multi-exponentiations and the verifier 84 pairings. See Appendix J for details.*

Theorem 6 (Security) *Our second P-signature construction is secure given HSDH and TDH and the security of the GS commitments and proofs.*

Consult Appendix K for proof.

References

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270, 2000.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 54–73, 2004.

[BBS04a] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures using strong Diffie-Hellman. In *CRYPTO*, volume 3152 of LNCS, pages 41–55, 2004.

[BBS04b] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures using strong diffie hellman. In *CRYPTO 2004*, LNCS. Springer Verlag, 2004.

[BCC04] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. Technical Report Research Report RZ 3450, IBM Research Division, March 2004.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Guiseppe Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, Illinois, 2–4 May 1988.

[BGdMM] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-Resistant Storage. Johns Hopkins University, CS Technical Report # TR-SP-BGMM-050705. `http://spar.isi.jhu.edu/~mgreen/correlation.pdf`, 2005.

[Bra93] Stefan Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI, April 1993.

[Bra99] Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy*. PhD thesis, Eindhoven Inst. of Tech. The Netherlands, 1999.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Vaudenay [Vau06], pages 427–444.

[BW07a] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.

[BW07b] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography*, pages 1–15, 2007.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *CRYPTO '90*, volume 403 of *LNCS*, pages 319–327, 1990.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CGH06] Sébastien Canard, Aline Gouget, and Emeline Hufschmitt. A handy multi-coupon system. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 66–81, 2006.

[Cha85] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 201–210, New York, NY, USA, 2006. ACM Press.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-Cash. In *EUROCRYPT*, volume 3494 of LNCS, pages 302–321, 2005.

[CHL06] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash. In *SCN (to appear)*, 2006.

[CL01] Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer Verlag, 2001.

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, volume 2576 of *LNCS*, pages 268–289, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72, 2004.

[CLM07] Jan Camenisch, Anna Lysyanskaya, and Mira Meyerovich. Endorsed e-cash. In *IEEE Symposium on Security and Privacy*, pages 101–115. IEEE Computer Society, 2007.

[CP93] David Chaum and Torben Pryds Pedersen. Transferred cash grows in size. In *EUROCRYPT '92*, volume 658 of *LNCS*, pages 390–407, 1993.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burt Kaliski, editor, *CRYPTO '97*, volume 1296 of *LNCS*, pages 410–424. Springer Verlag, 1997.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT '91*, volume 547 of *LNCS*, pages 257–265. Springer-Verlag, 1991.

[CVH02] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proc. 9th ACM Conference on Computer and Communications Security*. acm press, 2002.

[Dam90] Ivan Bjerre Damgård. Payment systems and credential mechanism with provable security against abuse by individuals. In Shafi Goldwasser, editor, *CRYPTO '88*, volume 403 of *LNCS*, pages 328–335. Springer Verlag, 1990.

[DDP05] Ivan Damgard, Kasper Dupont, and Michael Ostergaard Pedersen. Unclonable group identification. Cryptology ePrint Archive, Report 2005/170, 2005. `http://eprint.iacr.org/2005/170`.

[DDP06] Ivan Damgård, Kasper Dupont, and Michael Østergaard Pedersen. Unclonable group identification. In Vaudenay [Vau06], pages 555–572.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. *J. ACM*, 50(6):852–921, 2003.

[DSMP88] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *CRYPTO '87*, volume 293 of *LNCS*, pages 52–72. Springer-Verlag, 1988.

[EJA+04] Endre Bangerter, Jan Camenisch, Anna Lysyanskaya. In, Cambridge 12th International Workshop on Security Protocols 2004, and 2004. England, 26-28 April 2004. Springer Verlag. A Cryptographic Framework for the Controlled Release Of Certified Data. In *12th International Workshop on Security Protocols 2004*, Cambridge, England, 26 April 2004. Springer.

[FO98] E. Fujisaki and T. Okamoto. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of LNCS, pages 32–46. Springer, 1998.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer Verlag, 1987.

[FY93] Matthew Franklin and Moti Yung. Towards provably secure efficient electronic cash. In *proceedings of ICALP '93*, volume 700 of LNCS, pages 265–276, 1993.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a method of cryptographic protocol design. In *Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 174–187. IEEE Computer Society Press, 1986.

[GR04] S. Galbraith and V. Rotger. Easy decision diffie-hellman groups. *Journal of Computation and Mathematics*, 7:201–218, 2004.

[GS07] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. `http://eprint.iacr.org/2007/155`, 2007.

[GT03] Shafi Goldwasser and Yael Tauman. On the (in)security of the fiat-shamir paradigm. *Electronic Colloquium on Computational Complexity (ECCC)*, 10(015), 2003. `ftp://ftp.eccc.uni-trier.de/pub/eccc/reports/2003/TR03-015/index.html`. To appear in FOCS 2003.

[JS04] Stanislaw Jarecki and Vitaly Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. In *EUROCRYPT*, volume 3027 of *LNCS*, pages 590–608, 2004.

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In *EUROCRYPT*, pages 97–114, 2007.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT*, pages 143–154, 1996.

[Kat03] Jonathan Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2003.

[LRSW99] Anna Lysyanskaya, Ron Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *LNCS*, 1999.

[Lys02] Anna Lysyanskaya. *Signature Schemes and Applications to Cryptographic Protocol Design*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 2002.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

[NSN05] Lan Nguyen and Rei Safavi-Naini. Dynamic k-times anonymous authentication. In *ACNS 2005*, number 3531 in LNCS, pages 318–333. Springer Verlag, 2005.

[Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '92*, volume 576 of *LNCS*, pages 129–140, 1992.

[Sco02] Mike Scott. Authenticated id-based key exchange and remote log-in with insecure token and pin number. `http://eprint.iacr.org/2002/164`, 2002.

[SCP00] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Necessary and sufficient assumptions for non-interactive zero-knowledge proofs of knowledge for all NP relations. In Ugo Montanari, José P. Rolim, and Emo Welzl, editors, *Proc. 27th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1853 of *LNCS*, pages 451–462. Springer Verlag, 2000.

[TFS04] Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-times anonymous authentication (extended abstract). In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2004.

[TS06] Isamu Teranishi and Kazue Sako. -times anonymous authentication with a constant proving cost. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2006.

[Vau06] Serge Vaudenay, editor. *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.

[Ver04] Eric R. Verheul. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. *J. Cryptology*, 17(4):277–296, 2004.

[Wei05] Victor K. Wei. More compact e-cash with efficient coin tracing. Cryptology ePrint Archive, Report 2005/411, 2005. `http://eprint.iacr.org/`.

[Yao86] Andrew C. Yao. How to generate and exchange secrets. In *Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.

A Anonymous Credentials Based on P-Signatures

Recall that the participants in any credential system are users (who obtain credentials), organizations (who grant credentials) and a certification authority (CA). The existence of a CA allows users and organizations to register public keys. This is necessary, even if all other transactions are anonymous. Instead of saying "Alice has a credential" which, in the digital world, is not a well-formed statement, one needs to be able to say "The owner of pk_{Alice} (i.e. whoever knows sk_{Alice}has a credential." Then, so long as we believe that Alice does not reveal her secret key to other entities, there is a reason to believe that indeed it is Alice who has the credential. Here, it does not matter if we are talking about anonymous credentials or non-anonymous ones: even when we don't care about Alice's anonymity, unless users take steps to protect her secrets, a digital credentials system cannot be very meaningful.

An anonymous credential system consists of the following protocols:

Setup System parameters $params$ are generated, users and organizations generate their public and secret keys (pk, sk) and register their public keys with the CA. We will refer to PKI as the collection of all the public keys, and to the *identity* of the user as pk, his public key. As a result of this registration step, a user (whose private input is his secret key) obtains his root credential C_{CA}.

Pseudonym registration As a result of this protocol, a user and an organization agree on a pseudonym (nym) N for the user. The user's private input is his (sk, pk) and his C_{CA}; the organization does not have any private input. Their common output is N. The user's private output is $aux(N)$, some auxiliary information that may be needed later.

Credential issue As a result of this protocol, a user obtains a credential from an organization without revealing his identity, just based on his pseudonym N. The user U's private input to the protocol is his (sk_U, pk_U, aux_N), the organization's private input is its secret key sk_O, the user's private output is the credential C.

Proof of possession of a credential Here, a user who is known to one organization, O_1 under pseudonym N_1, and to another, O_2, under pseudonym N_2, and a credential C_1 from O_1, proves to O_2 that he has a credential from O_1. The user's private input to this protocol consists of $(sk_U, pk_U, P_1, aux_{N_1}, aux_{N_2}, C_1)$, while the values N_2 and pk_{O_1} are public. The organization verifies the proof.

An anonymous credential system should satisfy unforgeability and anonymity.

Informally, unforgeability requires that (1) corresponding to each pseudonym there is a well-defined identity and (2) if a user with pseudonym P successfully convinces an honest organization that she possesses a credential from another honest organization O', then it must be the case that organization O' has issued a credential to some pseudonym P' such that the identity of P' is the same as that of P.

Anonymity, informally, requires that, even an adversary that corrupts the CA and any subset of the organizations and users cannot distinguish the following two situations (1) it receives honestly generated public parameters, and is interfacing with honest users who obtain and show credentials as directed by the adversary; (2) it receives a different set of parameters, and is interfacing with users who obtain and show credentials as directed by the adversary, but instead of using the correct protocol for showing their credentials, they use a simulator algorithm that does not receive any inputs whose distribution depends on the identity of the user.

We now proceed to describe how an anonymous credential scheme can be constructed from P-signatures. Note that the reason that this scheme can be preferable to known schemes is that the proof of possession of a credential is non-interactive.

Suppose we are given a P-signature scheme. Then consider the following construction for an anonymous credential system:

Setup The system parameters $params$ are the parameters for the P-signature scheme. Note that they also include the parameters for Commit.

A user U's secret key sk_U will be chosen from the message space of the signature scheme (which coincides with the message space of the commitment scheme). The user's public key will be $pk_U = \mathsf{PublicKey}(sk_U)$ for an appropriately defined function $\mathsf{PublicKey}$.

Organizations (including the CA) will generate their key pairs using the key generation algorithm of the P-signature scheme.

The CA credential will be issued as follows:

1. The user forms his psedonym with the CA, $N_{CA} = \mathsf{Commit}(params, sk_U, opening)$ for an appropriately chosen $opening$. (Note that, since the commitment scheme is perfectly binding, this automatically guarantees that the identity associated with this pseudonym is well-defined.)
2. The user proves that he has committed to a sk such that his $pk_U = \mathsf{PublicKey}(sk)$ using an appropriate designated verifier [JSI96] non-malleable [Kat03] interactive proof.
3. The user and the CA run the protocol for obtaining a signature on a committed value (i.e. they run the ObtainSig and IssueSig protocols, respectively).

Pseudonym registration The user forms his pseudonym by forming a commitment to his secret key: for an appropriately chosen $opening$, $N = \mathsf{Commit}(params, sk_U, opening)$. (Again, since the commitment scheme is perfectly binding, this automatically gurantees that the identity associated with this pseudonym is well-defined.) The user proves that he has a credential from the CA for this pseudonym (as described below). The user then sends N to the organization and proves knowledge of $(sk, opening)$ using an appropriate designated verifier non-malleable interactive proof.[5]

The user's private output $aux(N) = opening$.

Credential issue The user U and the organization O run ObtainSig and IssueSig, respectively. The user's input is $(params, pk_O, sk_U, N, aux(N))$, while the organization's input is $(params, sk_O, N)$. As a result, the user obtains a signature σ on his sk_U, and so his credential is $C = \sigma$.

Proof of possession of a credential The user has a credential $C = \sigma_{O_1}(sk_U)$. He is known to organization O_2 as the owner of the pseudonym N. He needs to issue a non-interactive proof that a credential has been issued to the owner of N. This is done as follows:

1. Compute $(comm, \pi_1, opening) \leftarrow \mathsf{Prove}(params, pk_{O_1}, sk_U, C)$.
2. Compute $\pi_2 \leftarrow \mathsf{EqCommProve}(params, sk_U, opening, aux(N))$. (Where EqCommProve is explained in Section G.2. It is a non-interactive proof that the two commitments $comm$ and N are to the same value.)
3. Output $(N, comm, \pi_1, \pi_2)$.

We must now show that the resulting anonymous credentials scheme is secure.

Lemma 2 *The credentials scheme described above is unforgeable.*

Proof. (Sketch) Recall that the commitment scheme is perfectly binding. Therefore, corresponding to any setting of $params$, and any commitment N, there is exactly one value sk and opening $opening$ such that $N = \mathsf{Commit}(params, sk, opening)$, and exactly one corresponding value $pk = \mathsf{PublicKey}(sk)$. Therefore, with the pseudonym N, we can associate the identity pk, and (1) is satisfied. To satisfy (2), first suppose that the credential system is not unforgeable. Then we set up a reduction that breaks unforgeability of the P-signature scheme. Let F be the bijection that satisfies the unforgeability definition, and let ExtractSetup, Extract be the corresponding extractor. The reduction will be given $params$ as output by ExtractSetup, a public key pk, and access to a signing oracle. The reduction will make pk the public signing key of an arbitrary organization O under it's control. It will generate the keys for all other entities under its control correctly. Finally, it will make a random guess i that the adversary's ith proof of a credential O will be a forgery. Since the pseudonym registration protocol includes an (interactive) proof of knowledge of the opening to a commitment, every times the adversary wishes to register a pseudonym N with O, the values $(sk_A, opening)$ such that $N = \mathsf{Commit}(params, sk_A, opening)$ can be extracted using the knowledge extractor. Every time the adversary wishes to obtain a credential from an organization other than O, the reduction interacts with the adversary using the correct protocol. When the adversary, using pseudonym N, wishes to obtain a credential from O, the reduction already knows the values $(sk_A, opening)$ such that $N = \mathsf{Commit}(params, sk_A, opening)$. So it queries its signing oracle to obtain $\sigma \leftarrow \mathsf{Sign}(params, sk, sk_A)$, and then invokes SimIssue instead of IssueSig. (Note that SimIssue does not take any additional values, its simulation is based on rewinding the adversary.) The ith time the adversary produces a proof of possession of a credential from organization O consisting of $(N', comm, \pi_1, \pi_2)$, the reduction outputs π_1.

Now we analyze the reduction's probability of success. Note that the adversary's view is independent of i, O. If the reduction has guessed i, O correctly, and if the adversary's credential forgery is successful, then the identity defined by N' has not been granted a credential by O, but the credential proof will verify successfully. This means that $\mathsf{VerEqComm}(params, comm, N', \pi_2) = 1$ and $\mathsf{VerifyProof}(params, pk, comm, \pi_1) = 1$. Since (EqCommProve, VerEqComm) is perfectly sound, we know that $comm, N'$ are both commitments to the same value x. Since this is

[5] This ensures that, at registration time, the entity registering the pseudonym knows the secret key associated with this pseudonym, so that, for example, Alice could not get Bob to commit to his secret key and prove to her that he knows it, only to then have Alice use this commitment as her own pseudonym with another organization.

a forgery, we know O never issued a credential to the identity represented by $comm, N$, which means we have never queried our signing oracle on the commited value x. This means that when extractor extracts y, σ from $\pi, comm$, either $F^{-1}(y) \neq x$, or $\mathsf{VerifySig}(params, pk, F^{-1}(y), \sigma) = \mathsf{reject}$, or $\mathsf{VerifySig}(params, pk, F^{-1}(y), \sigma) = \mathsf{accept}$ and $F^{-1}(y) = x$ and $x \notin Q_{\mathsf{Sign}}$. In all cases, we break the unforgeability property.

Note that the reduction above implies unforgeability even as the adversarially controlled users talk to multiple organizations. However, each organization may only talk to one user at a time, because the reduction must extract the opening of the commitment (pseudonym) of the user wishing to obtain a credential from O, and it needs to rewind the adversary for that to happen. Similarly each organization must execute issue protocols sequentially. This is OK only if the adversary is never rewound to a point in time that happened before the last query to the signing oracle (because the signing oracle cannot be rewound). □

Lemma 3 *The credentials scheme described above is anonymous.*

Proof. (Sketch) Recall that we must show that no adversary can distinguish a real execution from one in which it is interfacing with users who, when obtaining and showing credentials do not use the correct protocols, but instead use a simulator algorithm that does not receive any inputs whose distribution depends on the identity of the user.

We now describe a series of hybrid experiments.

In hybrid experiment H_0, the adversary is interfacing with users and organizations carrying out the real protocols.

In hybrid experiment H_1, the parameters $params$ are generated using $\mathsf{SimSetup}(1^k)$. (Recall that $\mathsf{SimSetup}$ generates parameters for the commitment scheme that result in an information theoretically hiding commitment scheme.) Other than that, the adversary is interfacing with users and organizations carrying out the real protocols. The adversary's view in H_1 is indistinguishable from his view in H_0 because otherwise we could distinguish $params$ generated using Setup from those generated by $\mathsf{SimSetup}$.

In hybrid experiment H_2, the parameters $params$ and the value sim are generated using $\mathsf{SimSetup}$. The honest organizations with which the adversary is interfacing are carrying out the real protocols. The honest users will always form their pseudonyms correctly, but in the zero-knowledge proof of knowledge protocol that accompanies the registration (both the registration with the CA and the registration with other adversarial organizations), the users use the zero-knowledge simulator for that proof and not the actual proof protocol. Hybrid H_2 gives the adversary an indistinguishable view as that in hybrid H_1 because otherwise we contradict the zero-knowledge property of the zero-knowledge proof system.

Recall that, in addition to $params$, $\mathsf{SimSetup}$ also generates sim. The knowledge of sim is empowering in several important ways. The knowledge of sim allows one to (1) compute simulated proofs of equality of committed values (i.e. simulate $\mathsf{EqCommProve}$), and (2) simulate a proof that the committed value has been signed (recall the zero-knowledge part of our definition of P-signatures).

In hybrid experiment H_3, the only difference from H_2 is that honest users prove equality of committed values using the $\mathsf{SimEqCommProve}$ instead of using $\mathsf{EqCommProve}$. This should be indistinguishable from H_2 by the zero knowledge property of the P-signature.

In hybrid experiment H_4, the only difference from H_3 is that honest users generate proofs that committed values have been signed using $\mathsf{SimProve}$ instead of Prove. Note that this means they no longer have the opening of the resulting commitment $comm$. However, as we are now using $\mathsf{SimEqCommProve}$, we no longer need this opening. If this makes any difference to the adversary's view, then we again break the zero-knowledge property of the P-signature.

In hybrid experiment H_5, the only difference from H_4 is that honest users obtain signatures from adversarial organizations using $\mathsf{SimObtain}$ instead of $\mathsf{ObtainSig}$. (Note that they do not need to obtain the real signatures because they never use them, since their proofs that a commitment has been signed are always simulated.) If this makes any difference to the adversary's view, then it is easy to show that the user privacy part of the definition of security for P-signatures is broken.

In hybrid experiment H_6, the only difference from H_5 is that when honest users register pseudonyms, then commit to 1 instead of committing to their secret keys. Note that the view that the adversary gets as a result is the same as the view he gets in H_5, because the commitments are information-theoretically hiding, and all the proofs are simulated. Also note that in this experiment, the honest users run only protocols that never take users' identities as input. Therefore, we have obtained the desired simulator. □

B Secure Digital Signature

Definition 6 (Secure Signature Scheme [GMR88]) *We say that a signature scheme is secure (against adaptive chosen message attacks) if it is* Correct *and* Unforgeable.

Correctness. All signatures obtained using the Sign algorithm should be accepted by the VerifySig algorithm.

$$\forall msg \in \{0,1\}^* : \Pr[params_{Sig} \leftarrow \mathsf{SigSetup}(1^k); (pk, sk) \leftarrow \mathsf{Keygen}(params_{Sig}); \\ \sigma \leftarrow \mathsf{Sign}(params_{Sig}, sk, msg) : \mathsf{VerifySig}(params_{Sig}, pk, msg, \sigma) = 1] = 1$$

Unforgeability. No adversary should be able to output a valid message/signature pair (msg, σ) unless he has previously obtained a signature on msg. Formally, for every PPTM adversary $\mathcal{A}$, there exists a negligible function ν such that

$$\Pr[params_{Sig} \leftarrow \mathsf{SigSetup}(1^k); (pk, sk) \leftarrow \mathsf{Keygen}(params_{Sig}); \\ (Q_{\mathsf{Sign}}, msg, \sigma) \leftarrow \mathcal{A}(params_{Sig}, pk)^{\mathcal{O}_{\mathsf{Sign}}(params_{Sig}, sk, \cdot)} : \\ \mathsf{VerifySig}(params_{Sig}, pk, msg, \sigma) = 1 \wedge msg \notin Q_{\mathsf{Sign}}] < \nu(k).$$

$\mathcal{O}_{\mathsf{Sign}}(params_{Sig}, sk, msg)$ records all msg queries on Q_{Sign} and returns $\mathsf{Sign}(params_{Sig}, sk, msg)$.

C P-Signature Witness Indistinguishability.

No PPTM adversary can determine which of two message/signature pairs (σ_0, msg_0) and (σ_1, msg_1) was used to generate proof $(comm, \pi)$. Formally, for all PPTM adversaries $\mathcal{A}$, there exists a negligible function ν such that:

$$\Pr[params \leftarrow \mathsf{Setup}(1^k); (pk, \sigma_0, msg_0, \sigma_1, msg_1) \leftarrow \mathcal{A}(params); b \leftarrow \{0,1\}; \\ (comm, \pi) \leftarrow \mathsf{Prove}(params, pk, \sigma_b, msg_b) : \mathcal{A}(comm, \pi) = b \\ \wedge \mathsf{VerifySig}(params, pk, \sigma_0, msg_0) = 1 \wedge \mathsf{VerifySig}(params, pk, \sigma_1, msg_1) = 1] < 1/2 + \nu(k)$$

D Formal Assumptions

Boyen and Waters [BW07b] defined the Hidden SDH assumption over bilinear maps using symmetric groups $e : G \times G \rightarrow G_T$. We give a definition over asymmetric maps $e : G_1 \times G_2 \rightarrow G_T$. Note that in the symmetric setting, this is identical to the Boyen Waters HSDH assumption.

Definition 7 (Hidden SDH) *On input $g, g^x, u \in G_1$, $h, h^x \in G_2$ and $\{g^{1/(x+c_\ell)}, h^{c_\ell}, u^{c_\ell}\}_{\ell=1\ldots q}$, it is computationally infeasible to output a new tuple $(g^{1/(x+c)}, h^c, u^c)$. Formally, there exists a negligible function ν such that*

$$\Pr[(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BilinearSetup}(1^k); \\ u \leftarrow G_1; x, \{c_\ell\}_{\ell=1\ldots q} \leftarrow Z_p; \\ (A, B, C) \leftarrow \mathcal{A}(p, G_1, G_2, G_T, e, g, g^x, h, h^x, u, \{g^{1/(x+c_\ell)}, g^{c_\ell}, u^{c_\ell}\}_{\ell=1\ldots q}) : \\ (A, B, C) = (g^{1/(x+c)}, h^c, u^c) \wedge c \notin \{c_\ell\}_{\ell=1\ldots q}] < \nu(k).$$

We extend the HSDH assumption further and introduce a new assumption we call the Interactive HSDH assumption. We allow the adversary to adaptively query an oracle for HSDH triples on c_i of his choice.

Definition 8 (Interactive Hidden SDH assumption.) *No PPTM adversary can compute a tuple $(g^{1/(x+c)}, h^c, u^c)$ given (g, g^x, h, h^x, u) and permission to make q queries to oracle $\mathcal{O}_x(c)$ that returns $g^{1/(x+c)}$. The c used by the* *adversary must be different from the values it used to query $\mathcal{O}_x(\cdot)$. Formally, there exists a negligible function ν such that*

$$\Pr[(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BilinearSetup}(1^k); \\ x \leftarrow Z_p; u \leftarrow G_1; (A, B, C) \leftarrow \mathcal{A}^{\mathcal{O}_x(\cdot)}(p, G_1, G_2, G_T, e, g, g^x, h, h^x, u) : \\ \exists c \;:\; (A, B, C) = (g^{1/(x+c)}, h^c, u^c)] < \nu(k).$$

When $(p, G_1, G_2, G_T, e, g, h)$ and $H = h^x$ are fixed, we refer to tuples of the form $(g^{1/(x+c)}, h^c, u^c)$ as *HSDH tuples* (or, equivalently, as IHSDH tuples).

Note that we can determine whether (A, B, C) form an HSDH tuple using the bilinear map e, as follows: suppose we get a tuple (A, B, C). We check that $e(A, BH) = e(g, h)$ and that $e(u, B) = e(C, h)$.

We introduce a new assumption, we call the Triple DH.

Definition 9 (Triple DH) *On input $g, g^x, g^y, h, h^x, \{c_i, g^{1/(x+c_i)}\}_{i=1\ldots q}$, it is computationally infeasible to output a tuple $(h^{\mu x}, g^{\mu y}, g^{\mu xy})$. Formally, there exists a negligible function ν such that*

$$\Pr[(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BilinearSetup}(1^k); \\ (x, y) \leftarrow Z_p; \{c_i\}_{i=1\ldots q} \leftarrow Z_p; \\ (A, B, C) \leftarrow \mathcal{A}(p, G_1, G_2, G_T, e, g, g^x, g^y, h, h^x, \{c_i, g^{1/(x+c_i)}\}_{i=1\ldots q}) : \\ \exists \mu : (A, B, C) = (h^{\mu x}, g^{\mu y}, g^{\mu xy})] < \nu(k).$$

We also informally recall the DLIN and SXDH assumptions. These assumptions are needed for the Groth-Sahai pairing product equation proofs [GS07] (see Section 4).

Definition 10 (Decisional Linear Assumption [BBS04b]) *There exists a negligible function ν such that*

$$\Pr[(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BilinearSetup}(1^k); r, s \leftarrow Z_p; u, v, w \leftarrow G_1; \\ b \leftarrow \{0, 1\}; z_0 \leftarrow w^{s+t}; z_1 \leftarrow G_1 : \\ \mathcal{A}(p, G_1, G_2, G_T, e, g, h, u, v, w, u^r, v^s, z_b) = b] < 1/2 + \nu(k).$$

Definition 11 (External Diffie-Hellman Assumption (XDH)) *There exists a negligible function ν such that*

$$\Pr[(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BilinearSetup}(1^k); r, s \leftarrow Z_p; \\ b \leftarrow \{0, 1\}; z_0 \leftarrow g^{rs}, z_1 \leftarrow G_1 : \mathcal{A}(p, G, G_T, e, g, g^r, g^s, z_b) = b] < 1/2 + \nu(k).$$

The XDH assumption can be similarly defined to hold in G_2. The **SXDH** assumption states that XDH holds in both G_1 and G_2. The SXDH assumption was first used by Scott [Sco02], and has been discussed and used extensively since [BBS04b, GR04, Ver04, BGdMM].

E Instantiating GS Proofs using Modules

By expressing the implementation of non-interactive proofs in the language of modules Groth and Sahai can remain general with respect to possible instantiations of their proof system. Modules that fulfill the necessary requirements for their proofs exist both under the SXDH, the DLIN assumption, and with some restrictions the Subgroup Hiding assumption.

We follow Groth-Sahai: Let $(R, +, \cdot, 0, 1)$ be a commutative ring. An R module is a commutative group $(M, \cdot, 1)$, such that $\forall r, s \in R : \forall u, v \in M : u^{r+s} = u^r u^s \wedge (uv)^r = u^r v^r$.

Commitments. Commitments are realized using a Z_p module. For $u_1, \ldots, u_I$ elements of M, we call U the submodule of M generated by $u_1, \ldots, u_I$. To commit to $x \in G$, x is transformed into a unique element $x' \in M$ that for the perfectly binding setup is not element of U. For the perfectly hiding setup we create the parameters such that $U = M$. Now we commit by choosing $r_1, \ldots, r_I \in Z_p$ at random and computing

$$comm = x' \prod_{i=1}^{I} u_i^{r_i}.$$

NIZK Proofs The NIZK proofs require bilinear maps over modules. Let M_1, M_2, M_T be R modules. Then we define the bilinear map $E : M_1 \times M_2 \to M_T$. Let U generated by $u_1, \ldots, u_I$ be a submodule of M_1 and V generated by $v_1 \ldots v_I$ a submodule of M_2. The commitments to x_i and y_i are defined over M_1 and M_2 respectively.

In order to prove that

$$\mathsf{NIPK}\{(x_1, \ldots, x_Q, y_1, \ldots, y_Q) : \forall q : c_q = \mathsf{Commit}(x_q) \wedge d_q = \mathsf{Commit}(y_q) \wedge \prod_{q=1}^{Q} e(x_q, y_q) = t\},$$

the prover computes values π_i and ψ_i that fulfill the following verification equation:

$$\prod_{q=1}^{Q} E(c_q, d_q) = t' \prod_{i=1}^{I} E(u_i, \pi_i) E(\psi_i, v_i).$$

Where t' is a mapping of t to M_T. The values π_i and ψ_i can be computed from the x_i and y_i together with their commitments and opening information.

This will be made more concrete in our instantiation based on the SXDH assumption in Appendix L.

F Committing to group elements

In all of our constructions, we choose bilinear groups G_1, G_2, G_T with bilinear map e, and then use the GS commitments to commit to elements $x \in G_1 \cup G_2$. However, most of [GS07] focusses on commitments and proofs for elements of modules. Here we describe the techniques suggested by Groth and Sahai for using these commitments to commit to group elements. Using group elements instead of modules also allows us to get the extraction properties necessary for our construction.

We describe commitment to group elements in the SXDH and DLIN settings.

In the SXDH setting, for commitments to elements in G_1 (committing to G_2 is similar):

The parameters are generated by choosing random s, z and computing $u_1 = (g, g^z)$ and $u_2 = (g^s, g^{sz})$. The public parameters are u_1, u_2. If extraction is necessary, the trapdoor will be s, z.

GS describe commitments to elements in the module $M = G \times G$ as follows: To commit to element $X = (x_1, x_2) \in M$ choose random $r_1, r_2 \in Z_p$, and compute $X u_1^{r_1} u_2^{r_2}$ (where multiplication is entry-wise).

One can commit to $x \in G$ by choosing random $r_1, r_2 \in Z_p$ and computing $(1, x) u_1^{r_1} u_2 r_2$. Opening would reveal x, r_1, r_2. In this case, given the trapdoor s, z, we will be able to extract x from a commitment (c_1, c_2) by computing c_2 / c_1^z. Thus, this is perfectly binding and extractable.

Note that because all operations in the module M are entry-wise, any relationship that holds over elements $(1, x), (1, y) \in M$ will also hold over group element $x, y \in G$ [6]. GS proofs demonstrate that the proved relationship holds over any possible opening for the given commitments. Thus, it must hold for the unique $(1, x), (1, y)$ which are produced by the extraction algorithm described above, and as mentioned, this means the proved relationshipls must hold over group elements x, y.

Simulated parameters are generated by choosing random $s, z, w \in Z_p$ and computing $u_1 = (g, g^z)$ and $u_2 = (g^s, g^w)$. The public parameters will be u_1, u_2. The simulation trapdoor will be s, z, w. Note that these public parameters will be indistinguishable from those described above by SXDH.

[6] the bilinear map E over M is not entry-wise, but does still imply that any relationship over $E((1, x), (1, y))$ also holds over $e(x, y)$

Note that the resulting commitment scheme is perfectly hiding. Further, we can form commitments which are identical to those described above but for which we can use the simulation trpdoor to open to any value for which we know the discrete logaritm. We compute such a commitment by choosing random $c_1, c_2 \in Z_p$ and computing (g^{c_1}, g^{c_2}). To open a commitment this commitment to any value g^{ϕ}, we need only find a solution (r_1, r_2) to the equations $c_1 = r_1 + sr_2$ and $c_2 = \phi + zr_1 + wr_2$.

In the DLIN setting, for commitments to elements in G_1 (committing to G_2 is similar):

The parameters are generated by choosing random a, b, z, s and computing $u_1 = (g^a, 1, g)$ and $u_2 = (g^b, 1, g)$, and $u_3 = (g^{az}, g^{bs}, g^{z+s})$. The public parameters are u_1, u_2, u_3. If extraction is necessary, the trapdoor will be a, b, z, s.

GS describe commitments to elements in the module $M = G \times G$ as follows: To commit to element $X = (x_1, x_2, x_3) \in M$ choose random $r_1, r_2, r_3 \in Z_p$, and compute $X u_1^{r_1} u_2^{r_2} u_3^{r_3}$ (where multiplication is entry-wise).

One can commit to $x \in G$ by choosing random $r_1, r_2 r_3 \in Z_p$ and computing $(1, 1, x) u_1^{r_1} u_2 r_2 u_3$. Opening would reveal x, r_1, r_2, r_3. In this case, given the trapdoor a, b, s, z, we will be able to extract x from a commitment (c_1, c_2, c_3) by computing $c_3 / (c_1^{1/a} c_2^{1/b})$.

Note that again any relationship that holds over elements $(11, x), (1, 1, y) \in M$ will also hold over group element $x, y \in G$. Thus, we can using GS proofs on commitments to x, y to prove statements about x, y.

Simulated parameters are generated by choosing random $a, b, s, z, w \in Z_p$ and computing $u_1 = (g^a, 1, g)$ and $u_2 = (g^b, 1, g)$ and $u_3 = (g^{az}, g^{bs}, g^w)$. The public parameters will be u_1, u_2. The simulation trapdoor will be a, b, s, z. Note that these public parameters will be indistinguishable from those described above by DLIN.

Note that the resulting commitment scheme is perfectly hiding. Further, we can form commitments which are identical to those described above but for which we can use the simulation trpdoor to open to any value for which we know the discrete logaritm. We compute such a commitment by choosing random $c_1, c_2, c_3 \in Z_p$ and computing $(g^{c_1}, g^{c_2}, g^{c_3})$. To open a commitment this commitment to any value g^{ϕ}, we need only find a solution (r_1, r_2, r_3) to the equations $c_1 = ar_1 + azr_3$, $c_2 = br_2 + bsr_3$ and $c_3 = \phi + r_1 + r_2 + (z + s)r_3$.

G Extensions to the GS Proof system

G.1 Zero-Knowledge for the GS Proof System

Groth and Sahai [GS07] define a composable zero-knowledge NIPK for pairing product equation proofs. There exists a simulator $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimProve})$ with the following two properties:

1. $\mathsf{SimSetup}(params)$ outputs $params_{GS}'$ such that they are computationally indistinguishable from the output of $\mathsf{GSSetup}(params)$. Let $params_1' \in params_{GS}'$ be the parameters for the commitment scheme in G_1. Using $params_1'$, commitments are perfectly hiding – this means that for all commitments $comm$,

$$\forall x \in G, \exists opening : \mathsf{VerifyOpening}(params_1', comm, x, opening) = \mathsf{accept}.$$

2. Using the $params_{GS}'$ generated by the challenger, GS proofs become zero-knowledge. Suppose an unbounded adversary $\mathcal{A}$ generates a statement s consisting of the pairing product equations and a set of commitments $(c_1, \ldots, c_M, d_1, \ldots, d_N)$. The adversary outputs the statement s, and the opening of the commitments $W = (x_1, \ldots, x_M, y_1, \ldots, y_N, openings_0)$ such that they satisfy s. The challenger flips a coin to get $b \leftarrow \{0, 1\}$. If $b = 0$, then he outputs $\pi \leftarrow \mathsf{GSProve}(params_{GS}', s, W)$. If $b = 1$, then he outputs $\pi \leftarrow \mathsf{SimProve}(params_{GS}', s)$. The adversary gets π. The adversary guesses π with probability exactly $1/2$.

Groth and Sahai provide some useful tools for helping prove that particular GS proofs are zero-knowledge. Since GS proofs are witness indistinguishable, all a simulator has to do is come up with some witness for the equations. Witness indistinguishability guarantees that it is distributed identically to real witnesses. Groth and Sahai construct a $\mathsf{GSSimSetup}(params)$ function that outputs $params_{GS}$ that are (1) computationally indistinguishable from the output of $\mathsf{GSSetup}(params)$ and (2) allow us to open $comm = \mathsf{GSCommit}(G, b, \theta, opening)$ to any θ as long as we know $b, \theta, opening$. If a GS proof contains multiple pairing product equations, we can open $comm$ in a different way for each equation. Thus, we can have different witnesses for each equation. (This does not work for value $comm' = \mathsf{GSCommit}(G, x, opening)$).

G.2 Zero-Knowledge Proof of Equality of Committed Exponents

Suppose we know $c_1 = \mathsf{GSExpCommit}(G_1, g, \alpha)$ and $c_2 = \mathsf{GSExpCommit}(G_1, g, \alpha)$ as well as the openings to c_1 and c_2. We want to prove the statement

$$\mathsf{NIPK}\{((c_1 : g^\alpha), (c_2 : g^\beta)) : \alpha = \beta\}.$$

We calculate $d = \mathsf{GSExpCommit}(G_2, h, 1)$. Then we construct the proof

$$\pi \leftarrow \mathsf{NIPK}\{((c_1 : a), (c_2 : b), (d : h^\theta)) : e(a/b, h^\theta) = 1 \wedge e(g, h^\theta)e(1/g, h) = 1\}.$$

f-Extractability. We use $\mathsf{GSExtractSetup}(params)$ to generate $params_{GS}$ and a trapdoor td that lets us open all commitments. Suppose an adversary gives us a proof π. We extract $a = g^\alpha$, $b = g^\beta$, and $c = h^\theta$. By the soundness of the GS proof system, we have that $e(g, c)e(1/g, h) = 1$. This means $e(g, c) = e(g, h)$, so $c = h^1$. Thus we have $\theta = 1$. We can now transform the clause $e(a/b, h^\theta) = 1$ to $e(a/b, h) = 1$. Since e is non-degenerate, this means $a/b = 1$, and thus $\alpha = \beta$.

Composable Zero Knowledge. We need to construct $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimProve})$. We use the $\mathsf{GSSimSetup}$ algorithm provided by Groth and Sahai that outputs a trapdoor that allows us to open $\mathsf{GSExpCommit}(G, b, \theta, \mathit{opening})$ any way we want, as long as we know $b, \theta, \mathit{opening}$ (see Appendix G.1 above). We can open it to different values of θ in each pairing product equation.

The simulator gets as input c_1 and c_2. All the simulator needs to do is construct a witness for the individual equations of the proof

$$\pi \leftarrow \mathsf{NIPK}\{((c_1 : a), (c_2 : b), (d : h^\theta)) : e(a/b, h^\theta) = 1 \wedge e(g, h^\theta)e(1/g, h) = 1\}.$$

It sets $\theta = 0$ and computes $d = \mathsf{GSExpCommit}(G_2, h, 0, \mathit{opening})$. Thus, we satisfy the pairing product equation $e(a/b, h^\theta) = 1$ because $h^\theta = 1$. To satisfy the second pairing product equation, we open d to $\theta = 1$. Thus, we satisfy $e(g, h^\theta)e(1/g, h) = 1$. As a result, the simulator has a witness for the proof. By witness indistinguishability, the simulated witness is indistinguishable from real witnesses. Thus we get zero-knowledge.

H Weak Boneh-Boyen Signature

Theorem 7 *Let $F(x) = (h^x, u^x)$, where $u \in G_1$ and $h \in G_2$ as given in the statement of the IHSDH assumption. The Weak Boneh-Boyen signature scheme is F-secure given IHSDH.*

Proof. The proof of security is trivial given the IHSDH assumption. Correctness is straightforward. To prove unforgeability, we create a reduction to the IHSDH assumption. The reduction gets as input $(p, G_1, G_2, G_T, e, g, G, h, H, u)$, where $G = g^x$ and $H = h^x$ for some secret x. The reduction sets up the public parameters of the Boneh Boyen signature scheme $params = ((p, G_1, G_2, G_T, e, g, h)$ and a public-key $pk = (H, G)$. To answer a signature query on message msg_ℓ, the reduction sends a query to $\mathcal{O}_w(msg_\ell)$ and sends $g^{1/(x+msg_\ell)}$ back to the adversary. Eventually, the adversary will output a forgery (σ, y), where $\sigma = g^{1/(x+m)}$, $y = F(m) = (h^m, u^m)$, and $m \neq msg_\ell$ for all ℓ. The reduction can then output the IHSDH tuple (σ, h^m, u^m). □

I Security of new signature scheme

Theorem 8 *Let $F(x) = (h^x, u^x)$, where $u \in G_1$ and $h \in G_2$ as in the HSDH and TDH assumptions. Our new signature scheme is F-secure given HSDH and TDH.*

Proof. Correctness is straightforward. *Unforgeability* is more difficult. Suppose we try to do a straightforward reduction to HSDH. The reduction will setup the parameters for the signature scheme. Whenever the adversary queries $\mathcal{O}_{\mathsf{Sign}}$, the reduction will use one of the provided tuples $(g^{1/(x+c_\ell)}, g^{c_\ell}, v^{c_\ell})$ to construct a signature for input message msg_ℓ. We choose r_ℓ such that $c_\ell = msg_\ell + \beta r_\ell$. Thus, $C_1 = g^{1/(x+c_\ell)}$, $C_2 = w^{r_\ell}$ and $C_3 = u^{r_\ell}$. (The acual proof will be more complicated, because we don't know c_ℓ and therefore cannot calculate r_ℓ directly).

Eventually, the adversary returns $F(m) = (h^m, u^m)$ and a valid signature (C_1, C_2, C_3). Since the signature is valid, we get that $C_1 = g^{1/(x+m+\beta r)}$, $C_2 = w^r = h^{\beta r}$, and $C_3 = u^r$. We can have two types of forgeries. In Type 1, the adversary returns a forgery such that $m + \beta r \neq msg_\ell + \beta r_\ell$ for all of the adversary's previously queried messages msg_ℓ, in which case we can easily create a new HSDH tuple. In Type 2, the adversary returns a forgery such that $m + \beta r = msg_\ell + \beta r_\ell$. In this case, we cannot use the forgery to construct a new HSDH tuple. Therefore, we divide our proof into two categories. In Type 1, we reduce to the HSDH assumption. In Type 2, we reduce to the TDH assumption.

Type 1 forgeries: $\beta r + m \neq \beta r_\ell + msg_\ell$ for any r_ℓ, msg_ℓ from a previous query. The reduction gets an instance of the HSDH problem $(p, G_1, G_2, G_T, e, g, v, \tilde{v}, h, u, \{C_\ell, H_\ell, U_\ell\}_{\ell=1\ldots q})$, such that $v = h^x$ and $\tilde{v} = g^x$ for some unknown x, and for all ℓ, $C_\ell = g^{1/(x+c_\ell)}$, $H_\ell = h^{c_\ell}$, and $U_\ell = u^{c_\ell}$ for some unknown c_ℓ. The reduction sets up the parameters of the new signature scheme as $(p, G_1, G_2, e, g, h, u, z = e(g, h))$. Next, the reduction chooses $\beta \leftarrow Z_p$ and calculates $w = h^\beta, \tilde{w} = g^\beta$. The reduction gives the adversary the public parameters and the public-key $(v, w, \tilde{v}, \tilde{w})$.

Suppose the adversary's ℓth query is to Sign message msg_ℓ. The reduction will implicitly set r_ℓ to be such that $c_\ell = msg_\ell + \beta r_\ell$. This is an equation with two unknowns, so we do not know r_ℓ and c_ℓ. The reduction sets $C_1 = C_\ell$. It computes $C_2 = H_\ell / h^{msg_\ell} = h^{c_\ell}/h^{msg_\ell} = w^{r_\ell}$. Then it computes $C_3 = (U_\ell / u^{msg_\ell})^{1/\beta} = (u^{c_\ell}/u^{msg_\ell})^{1/\beta} = u^{(c_\ell - msg_\ell)/\beta} = u^{r_\ell}$. The reduction returns the signature (C_1, C_2, C_3).

Eventually, the adversary returns $F(m) = (F_1, F_2)$ and a valid signature (C_1, C_2, C_3). Since this is a valid F-forger, we get that $F_1 = h^m$, $F_2 = u^m$ and $C_1 = g^{1/(x+m+\beta r)}$, $C_2 = w^r = h^{\beta r}$, and $C_3 = u^r$. Since this is a Type 1 forger, we also have that $m + \beta r \neq msg_\ell + \beta r_\ell$ for any of the adversary's previous queries. Therefore, $(C_1, F_1 C_2, F_2 C_3^\beta) = (g^{1/(x+m+\beta r)}, h^{m+\beta r}, u^{m+\beta r})$ is a new HSDH tuple.

Type 2 forgeries: $\beta r + m = \beta r_\ell + msg_\ell$ for some r_ℓ, msg_ℓ from a previous query. The reduction receives $(p, G_1, G_2, G_T, e, g, h, X, Z, Y, \{\sigma_\ell, c_\ell\})$, where $X = h^x$, $Z = g^x$, $Y = g^y$, and for all ℓ, $\sigma_\ell = g^{1/(x+c_\ell)}$. The reduction chooses $\gamma \leftarrow Z_p$ and sets $u = Y^\gamma$. The reduction sets up the parameters of the new signature scheme as $(p, G_1, G_2, e, g, h, u, z = e(g, h))$. Next the reduction chooses $\alpha \leftarrow Z_p$, and calculates $v = h^\alpha$, $w = X^\gamma$, $\tilde{v} = g^\alpha$, $\tilde{w} = Z^\gamma$. It gives the adversary the parameters and the public-key $(v, w, \tilde{v}, \tilde{w})$. Note that we set up our parameters and public-key so that $\beta = x\gamma$, for some unknown β and $u = g^{\gamma y}$.

Suppose the adversary's ℓth query is to Sign message msg_ℓ. The reduction sets $r_\ell = (\alpha + msg_\ell)/(c_\ell \gamma)$ (which it can compute). The reduction computes $C_1 = \sigma_\ell^{1/(\gamma r_\ell)} = (g^{1/(x+c_\ell)})^{1/(\gamma r_\ell)} = g^{1/(\gamma r_\ell (x+c_\ell))} = g^{1/(\alpha + msg_\ell + \beta r_\ell)}$. Since the reduction knows r_ℓ, it computes $C_2 = w^{r_\ell}$, $C_3 = u^{r_\ell}$ and send (C_1, C_2, C_3) to $\mathcal{A}$.

Eventually, the adversary returns $F(m) = (F_1, F_2)$ and a valid signature (C_1, C_2, C_3). Since this is an F-forgery, we get that $F_1 = h^m$, $F_2 = u^m$ and that $C_1 = g^{1/(x+m+\beta r)}$, $C_2 = w^r = h^{\beta r}$, and $C_3 = u^r$. Since this is a Type 2 forger, we also have that $m + \beta r = msg_\ell + \beta r_\ell$ for one of the adversary's previous queries. (We can learn m_ℓ and r_ℓ by comparing $F_1 C_2$ to $h^{m_\ell} w^{r_\ell}$ for all ℓ.) We define $\delta = m - msg_\ell$. Since $m + \beta r = msg_\ell + \beta r_\ell$, we also get that $\delta = \beta(r_\ell - r)$. Using $\beta = x\gamma$, we get that $\delta = x\gamma(r_\ell - r)$. We compute: $A = F_1 / h^{m_\ell} = h^{m - m_\ell} = h^\delta$, $B = u^{r_\ell}/C_3 = u^{r_\ell - r} = u^{\delta/\gamma x} = g^{y\delta/x}$ and $C = (F_2/u^{m_\ell})^{1/\gamma} = u^{(m-m_\ell)/\gamma} = u^{\delta/\gamma} = g^{\delta y}$. We implicitly set $\mu = \delta/x$, thus $(A, B, C) = (h^{\mu x}, g^{\mu y}, g^{\mu x y})$ is a valid TDH tuple. $\square$

J Efficiency

We begin by giving formulas for calculating the efficiency of GS pairing product equation proofs, in both the SXDH and the DLIN setting.

First, recall that commitments based on SXDH consist of 2 elements in G, while those based on DLIN setting require 3 elements in G. Our formulas will include the cost of including the commitments that make up the statement.

Suppose a pairing product equation has a pairing of the form $e(a_q, b_q)$. We call this a 'constant pairing', it arises when $\forall m, n : \alpha_{q,m} = \beta_{q,n} = 0$. We call all pairings that are not constant 'non-constant pairings'. Let $\hat{Q}$ be total number of non-constant pairings in a set of pairing product equations. We calculate the efficiency of proving/verifying a set of L pairing product equations, over M variables in G_1, N variables in G_2, and $\hat{Q}$ non-constant pairings.

In the SXDH setting, a proof for a set of pairing product equations as defined above would consist of $4L + 2M$ elements of G_1 and $4L + 2N$ elements of G_2 (this includes the commitments c_m and d_n). Calculating each group element requires a single multi-base exponentiation. The verifier must perform $16L + 4\hat{Q}$ pairings. The DLIN setting is symmetric, so $G_1 = G_2$, and we only have variables x_m. The proof consists of $9L + 3M$ elements of G_1, each of which requires a single multi-base exponentiation. The verifier performs $18L + 6\hat{Q}$ pairings.

Theorem 9 (Efficiency of our First Construction) *Using SXDH, each P-signature proof for the weak Boneh-Boyen signature scheme consists of* 12 *elements in* G_1 *and* 10 *elements in* G_2*. The prover performs* 22 *multi-exponentiations and the verifier* 44 *pairings. Using DLIN, each P-signature proof consists of* 27 *elements in* $G_1 = G_2$*. The prover performs* 27 *multi-exponentiations and the verifier* 54 *pairings.*

Proof. We rewrite the proof as

$$\pi = \mathsf{NIPK}\{((M_h : h^\alpha), (M_u : u^\beta), (\Sigma : x)) : e(u, h^\alpha) \cdot e(u^\beta, h^{-1}) = 1 \wedge e(x, vh^\alpha) = z)\}.$$

We have $L = 2$ equations, $M = 2$ variables in G_1, $N = 1$ variable in G_2, and $\hat{Q} = 3$ non-constant pairings. SXDH requires $4L + 2M = 12$ elements in G_1 and $4L + 2N = 10$ elements in G_2 (Each group element can be calculated using one multi-exponentiation); the verifier performs $16L + 4\hat{Q} = 44$ pairings. With $M' = M + N = 3$ variables in $G_1 = G_2$ DLIN requires $9L + 3M' = 27$ elements in G_1; the verifier performs $18L + 6\hat{Q} = 54$ pairings. □

See Appendix L for details on the construction.

Theorem 10 (Efficiency of our Second Construction) *Using SXDH GS proofs, each P-signature proof for our new signature scheme consists of 18 elements in* G_1 *and 16 elements in* G_2*. The prover performs 34 multi-exponentiation and the verifier 68 pairings. Using DLIN, each P-signature proof consists of* 42 *elements in* $G_1 = G_2$*. The prover has to do* 42 *multi-exponentiations and the verifier* 84 *pairings.*

Proof. The non-interactive proof can be rewritten as

$$\pi = \mathsf{NIPK}\{((\Sigma : C_1), (R_w : C_2), (R_u : C_3)(M_h : h^\alpha), (M_u : u^\beta)) : \\ e(C_1, vh^\alpha C_2) = z \wedge e(u, C_2) \cdot e(C_3, w^{-1}) = 1 \wedge e(u, h^\alpha) \cdot e(u^\beta, h^{-1}) = 1\}.$$

We have $L = 3$, $M = 3$, and $N = 2$. The number of non-constant pairings $\hat{Q} = 5$. SXDH requires $4L + 2M = 18$ elements in G_1 and $4L + 2N = 16$ elements in G_1 (Each group element can be calculated using one multi-exponentiation). The verifier performs $16L + 4\hat{Q} = 68$ pairings. With $M' = M + N = 5$ variables in $G_1 = G_2$ DLIN requires $9L + 3M' = 42$ elements in G_1; the verifier performs $18L + 6\hat{Q} = 84$ pairings. □

K Security of Second Construction of P-Signatures

Theorem 11 (Security) *Our second P-signature construction is secure given HSDH and TDH and the security of the GS commitments and proofs.*

Proof. Correctness. VerifyProof will always accept properly formed proofs.

Signer Privacy. We must construct the SimIssue algorithm that is given as input $params$, a commitment $comm$ and a signature $\sigma = (C_1, C_2, C_3)$ and must simulate the adversary's view. SimIssue will invoke the simulator for the two-party computation protocol. Recall that in two-party computation, the simulator can first extract the input of the adversary: in this case, some $(\rho_1, \rho_2, msg, opening)$. Then SimIssue checks that $comm = \mathsf{Commit}(params, msg, opening)$; if it isn't, it terminates. Otherwise, it sends to the adversary the values $(C_1' = C_1^{1/\rho_2}, C_2' = C_2^{1/\rho_1}, C_3' = C_3^{1/\rho_1})$. Suppose the adversary can determine that it is talking with a simulator. Then it must be the case that the adversary's input to the protocol was incorrect which breaks the security properties of the two-party computation.

User Privacy. The simulator will invoke the simulator for the two-party computation protocol. Recall that in two-party computation, the simulator can first extract the input of the adversary (in this case, some (α', β'), not necessarily the valid secret key). Then the simulator is given the target output of the computation (in this case, the value x which is just a random value that the simulator can pick itself), and proceeds to interact with the adversary such that if the adversary completes the protocol, its output is x. Suppose the adversary can determine that it is talking with a simulator. Then it breaks the security of the two-party computation protocol.

Zero knowledge. Consider the following algorithms. SimSetup runs BilinearSetup to get $params_{BM} = (p, G_1, G_2, G_T, e, g, h)$. It then picks $t \leftarrow Z_p$ and sets up $u = g^a$. Next it calls GSSimSetup($params_{BM}$) to obtain $params_{GS}$ and sim. The final parameters are $params = (params_{GS}, u, z = e(g, h))$ and $sim = (a, sim)$. Note that the distribution of $params$ is indistinguishable from the distribution output by Setup. SimProve receives $params$, sim, and public key $(v, \tilde{v}, w, \tilde{w})$ and can use trapdoor sim to create a random P-signature forgery in SimProve as follows. Pick $s, r \leftarrow Z_p$ and compute $\sigma = g^{1/s}$. We implicitly set $msg = s - \alpha - r\beta$. Note that the simulator does not know msg and α. However, he can compute $h^{msg} = h^s/(vw^r)$ and $u^{msg} = u^s/(\tilde{v}^a \tilde{w}^{ar})$. Now he can use σ, h^{msg}, u^{msg}, w^r, u^r as a witness and construct the proof π in the same way as the real Prove protocol. By the witness indistinguishability of the GS proof system, a proof using the faked witnesses is indistinguishable from a proof using a real witness, thus SimProve is indistinguishable from Prove.

Finally, we need to show that we can simulate proofs of EqCommProve given the trapdoor sim_{GS}. This follows from composable zero knoweldge of EqCommProve. See Appendix G.

Unforgeability. Consider the following algorithms: ExtractSetup(1^k) outputs the usual $params$, except that it invokes GSExtractSetup to get alternative $params_{GS}$ and the trapdoor $td = (td_1, td_2)$ for extracting GS commitments in G_1 and G_2. The parameters generated by GSSetup are indistinguishable from those generated by GSExtractSetup, so we know that the parameters generated by ExtractSetup will be indistinguishable from those genrated by Setup.

Extract($params, td, comm, \pi$) extracts the values from commitment $comm$ and the commitments M_h, M_u contained in the proof π using the GS commitment extractor. If VerifyProof accepts then $comm = M_h$. Let $F(msg) = (h^{msg}, u^{msg})$.

Now suppose we have an adversary that can break the unforgeability of our P-signature scheme for this extractor and this bijection.

A P-signature forger outputs a proof from which we extract $(F(m), \sigma)$ such that either (1) VerifySig($params, pk, m, \sigma$) = reject, or (2) $comm$ is not a commitment to m, or (3) the adversary never queried us on m. Since VerifyProof checks a set of pairing product equations, f-extractability of the GS proof system trivially ensures that (1) never happens. Since VerifyProof checks that $M_h = comm$, this ensures that (2) never happens. Therefore, we consider the third possibility. The extractor calcualtes $F(m) = (h^m, u^m)$ where m is fresh. Due to the randomness element r in the signature scheme, we have two types of forgeries. In a Type 1 forgery, the extractor can extract from the proof a tuple of the form $(g^{1/(\alpha+m+\beta r)}, w^r, u^r, h^m, u^m)$, where $m + r\beta \neq msg_\ell + r_\ell\beta$ for any (msg_ℓ, r_ℓ) used in answering the adversary's signing or proof queries. The second type of forgery is one where $m + r\beta = msg_\ell + r_\ell\beta$ for (msg_ℓ, r_ℓ) used in one of these previous queries. We show that a Type 1 forger can be used to break the HSDH assumption, and a Type 2 forger can be used to break the TDH assumption.

Type 1 forgeries: $\beta r + m \neq \beta r_\ell + msg_\ell$ for any r_ℓ, msg_ℓ from a previous query. The reduction gets an instance of the HSDH problem $(p, G_1, G_2, G_T, e, g, X, \tilde{X}, h, u, \{C_\ell, H_\ell, U_\ell\}_{\ell=1\ldots q})$, such that $X = h^x$ and $\tilde{X} = g^x$ for some unknown x, and for all ℓ, $C_\ell = g^{1/(x+c_\ell)}$, $H_\ell = h^{c_\ell}$, and $U_\ell = u^{c_\ell}$ for some unknown c_ℓ. The reduction sets up the parameters of the new signature scheme as $(p, G_1, G_2, e, g, h, u, z = e(g, h))$. Next, the reduction chooses $\beta \leftarrow Z_p$, sets $v = X, \tilde{v} = \tilde{X}$ and calculates $w = h^\beta, \tilde{w} = g^\beta$. The reduction gives the adversary the public parameters and the public-key $(v, w, \tilde{v}, \tilde{w})$.

Suppose the adversary's ℓth query is to Sign message msg_ℓ. The reduction will implicitly set r_ℓ to be such that $c_\ell = msg_\ell + \beta r_\ell$. This is an equation with two unknowns, so we do not know r_ℓ and c_ℓ. The reduction sets $C_1 = C_\ell$. It computes $C_2 = H_\ell/h^{msg_\ell} = h^{c_\ell}/h^{msg_\ell} = w^{r_\ell}$. Then it computes $C_3 = (U_\ell)^{1/\beta}/u^{msg_\ell/\beta} = (u^{c_\ell})^{1/\beta}/u^{msg_\ell/\beta} = u^{(c_\ell - msg_\ell)/\beta} = u^{r_\ell}$ The reduction returns the signature (C_1, C_2, C_3).

Eventually, the adversary returns a proof π. Since π is f-extractable and perfectly sound, we extract $\sigma = g^{1/(x+m+\beta r)}$, $a = w^r$, $b = u^r$, $c = h^m$, and $d = u^m$. Since this is a P-signature forgery, $(c, d) = (h^m, u^m) \notin F(Q_{\mathsf{Sign}})$. Since this is a Type 1 forger, we also have that $m + \beta r \neq msg_\ell + \beta r_\ell$ for any of the adversary's previous queries. Therefore, $(\sigma, ca, db^\beta) = (g^{1/(x+m+\beta r)}, h^{m+\beta r}, u^{m+\beta r})$ is a new HSDH tuple.

Type 2 forgeries: $\beta r + m = \beta r_\ell + msg_\ell$ for some r_ℓ, msg_ℓ from a previous query. The reduction receives $(p, G_1, G_2, G_T, e, g, h, X, Z, Y, \{\sigma_\ell, c_\ell\})$, where $X = h^x$, $Z = g^x$, $Y = g^y$, and for all ℓ, $\sigma_\ell = g^{1/(x+c_\ell)}$. The reduction chooses $\gamma \leftarrow Z_p$ and sets $u = Y^{\gamma}$. The reduction sets up the parameters of the new signature scheme as $(p, G_1, G_2, e, g, h, u, z = e(g, h))$. Next the reduction chooses $\alpha \leftarrow Z_p$, and calculates $v = h^{\alpha}$, $w = X^{\gamma}$, $\tilde{v} = g^{\alpha}$, $\tilde{w} = Z^{\gamma}$. It gives the adversary the parameters and the public-key $(v, w, \tilde{v}, \tilde{w})$. Note that we set up our parameters and public-key so that β is implicitly defined as $\beta = x\gamma$, and $u = g^{\gamma y}$.

Suppose the adversary's ℓth query is to Sign message msg_ℓ. The reduction sets $r_\ell = (\alpha + msg_\ell)/(c_\ell \gamma)$ (which it can compute). The reduction computes $C_1 = \sigma_\ell^{1/(\gamma r_\ell)} = (g^{1/(x+c_\ell)})^{1/(\gamma r_\ell)} = g^{1/(\gamma r_\ell (x+c_\ell))} = g^{1/(\alpha + msg_\ell + \beta r_\ell)}$. Since the reduction knows r_ℓ, it computes $C_2 = w^{r_\ell}$, $C_3 = u^{r_\ell}$ and send (C_1, C_2, C_3) to $\mathcal{A}$.

Eventually, the adversary returns a proof π. The proof π is f-extractable and perficetly sound, the reduction can extract $\sigma = g^{1/(x+m+\beta r)}$, $a = w^r$, $b = u^r$, $c = h^m$, and $d = u^m$. Therefore, $\mathsf{VerifySig}$ will always accept $m = F^{-1}(c, d)$, σ, a, b. We also know that if this is a forgery, then $\mathsf{VerifyProof}$ accepts, which means that $comm = M_h$, which is a commitment to m. Thus, since this is a P-signature forgery, it must be the case that $(c, d) = (h^m, u^m) \notin F(Q_{\mathsf{Sign}})$. However, since this is a Type 2 forger, we also have that $\exists \ell : m + \beta r = msg_\ell + \beta r_\ell$, where msg_ℓ is one of the adversary's previous Sign or Prove queries. We implicitly define $\delta = m - msg_\ell$. Since $m + \beta r = msg_\ell + \beta r_\ell$, we also get that $\delta = \beta(r_\ell - r)$. Using $\beta = x\gamma$, we get that $\delta = x\gamma(r_\ell - r)$. We compute: $A = c/h^{m_\ell} = h^{m - m_\ell} = h^{\delta}$, $B = u^{r_\ell}/b = u^{r_\ell - r} = u^{\delta/(\gamma x)} = g^{y\delta/x}$ and $C = (d/u^{m_\ell})^{1/\gamma} = u^{(m-m_\ell)/\gamma} = u^{\delta/\gamma} = g^{\delta y}$. We implicitly set $\mu = \delta/x$, thus $(A, B, C) = (h^{\mu x}, g^{\mu y}, g^{\mu x y})$ is a valid TDH tuple. □

L Instantiation of P-Signatures using SXDH

L.1 Instantiating Groth-Sahai Proofs using SXDH [GS07]

We review the Groth-Sahai [GS07] witness indistinguishable proofs based on the SXDH assumption. Let G be a group of prime order p. Then $M = G \times G$ is a module over the ring Z_p, with the operation being entry-wise multiplication: $\forall (a, b), (x, y) \in M : (a, b) \cdot (x, y) = (ax, by)$. If $e : G_1 \times G_2 \to G_T$ is a bilinear map, over groups G_1, G_2, and G_T of prime order p, we can construct modules $M_1 = G_1 \times G_1$, $M_2 = G_2 \times G_2$, and $M_T = G_T \times G_T \times G_T \times G_T$ using entry-wise multiplication. We define a bilinear map $E : M_1 \times M_2 \to M_T$ between modules as

$$E((a, b), (x, y)) = \big(e(a, x), e(a, y), e(b, x), e(b, y)\big)$$

$\mathsf{GSSetup}(p, G_1, G_2, G_T, e, g, h)$**.** Choose $z_1, z_2, s_1, s_2 \leftarrow Z_p$. We set $u_1 = (g, g^{z_1})$ and $u_2 = u_1^{s_1} = (g^{s_1}, g^{z_1 s_1})$ for commitments in G_1. Similarly we set $v_1 = (h, h^{z_2})$ and $v_2 = v_1^{s_2} = (g^{s_2}, g^{z_2 s_2})$ for commitments in G_2. Output the common random string $(p, params_1 = (G_1, g, u_1, u_2), params_2 = (G_2, h, v_1, v_2), G_T, e)$.

$\mathsf{Commit}(params_i, x, (r_1, r_2))$**.** Let $params_i$ be (G_1, g, u_1, u_2), w.l.o.g. . Use opening information $r_1, r_2 \in Z_p$ to output $comm = (1, x) u_1^{r_1} u_2^{r_1}$.

$\mathsf{GSExpCommit}(params_i, b, \theta, opening)$**.** Here we depart from Groth-Sahai. Let b be a base of the correct group. We compute $comm$ and output $(b, comm)$, where $comm = \mathsf{Commit}(params_i, (1, b^{\theta}), opening)$. [7]

$\mathsf{VerifyOpening}(params_i, comm, x, (r_1, r_2))$**.** Let $params_i$ be (G_1, g, u_1, u_2), w.l.o.g. . Output accept if $comm = (1, x) u_1^{r_1} u_2^{r_1}$. Note that a commitment $(b, comm)$ to exponent θ with opening $opening$ can be verified using $\mathsf{VerifyOpening}(params_i, comm, b^{\theta}, opening)$.

$\mathsf{GSProve}(params_{GS}, s, (\{x_m\}_{m=1,\dots,M}, \{y_n\}_{n=1,\dots,N}), (\{r_{m1}, r_{m2}\}_{m=1,\dots,M}, \{s_{n1}, s_{n2}\}_{n=1,\dots,M})$**.** A true statement s contains commitments

$$c_m = \mathsf{Commit}(params_1, x_m, (r_{m1}, r_{m2})) = (1, x_m) u_1^{r_{m1}} u_2^{r_{m2}} \text{ and}$$
$$d_n = \mathsf{Commit}(params_2, y_n, (s_{n1}, s_{n2})) = (1, y_n) v_1^{s_{n1}} v_2^{s_{n2}},$$

[7] Groth and Sahai use opening $r \leftarrow Z_p$ and output $comm = u_1^{\theta} u_2^{r}$. This is more efficient than our construction, but gives up some flexibility, e.g. in choosing the base b.

and the following values $\{a_q\}_{q=1\ldots Q} \in G_1$, $\{b_q\}_{q=1\ldots Q} \in G_2$, $t \in G_T$, and $\{\alpha_{q,m}\}_{m=1\ldots M, q=1\ldots Q} \in Z_p$, $\{\beta_{q,n}\}_{n=1\ldots N, q=1\ldots Q} \in Z_p$ such that the pairing product equation in the following proof description holds.

$$NIPK\{((c_1 : x_1), \ldots, (c_M : x_M), (d_1 : y_1), \ldots, (d_N : y_N)) : \prod_{q=1}^{Q} e(a_q \prod_{m=1}^{M} x_m^{\alpha_{q,m}}, b_q \prod_{n=1}^{N} y_n^{\beta_{q,n}}) = t.$$

Note that the commitments are homomorphic, i.e., the product of two commitments (and the sum of the openings) is a commitment (opening) to the product of the committed values:

$$(1, x_1)u_1^{r_{11}}u_2^{r_{12}} \cdot (1, x_2)u_1^{r_{21}}u_2^{r_{22}} = (1, x_1x_2)u_1^{r_{11}+r_{21}}u_2^{r_{12}+r_{22}}.$$

Constants a are treated as commitments with 0 opening. Similarly, the power of a commitment to the α (and the product of the opening with α) is a commitment to the committed value to the α:

$$((1, x_1)u_1^{r_{11}}u_2^{r_{12}})^{\alpha} = (1, x_1^{\alpha})u_1^{r_{11}\alpha}u_2^{r_{12}\alpha}$$

We use this property to compute shadow commitments

$$\tilde{c}_q = (1, \tilde{x_q} = a_q \prod_{m=1}^{M} x_m^{\alpha_{q,m}})u_1^{\tilde{r}_{m1}}u_2^{\tilde{r}_{m2}}, \quad \tilde{d}_n = (1, \tilde{y_q} = b_q \prod_{n=1}^{N} y_n^{\beta_{q,n}})v_1^{\tilde{s}_{n1}}v_2^{\tilde{s}_{n2}}.$$

We choose $\{t_{i,j}\}_{i,j=1\ldots 2} \leftarrow Z_p$ and compute

$$\pi_1 = v_1^{t_{11}}v_2^{t_{12}} \prod_{q=1}^{Q} \tilde{d}_q^{\tilde{r}_{q1}} \qquad \psi_1 = u_1^{-t_{11}}u_2^{-t_{21}} \prod_{q=1}^{Q} (1, \tilde{x}_q)^{\tilde{s}_{q1}}$$

$$\pi_2 = v_1^{t_{21}}v_2^{t_{22}} \prod_{q=1}^{Q} \tilde{d}_q^{\tilde{r}_{q2}} \qquad \psi_2 = u_1^{-t_{12}}u_2^{-t_{22}} \prod_{q=1}^{Q} (1, \tilde{x}_q)^{\tilde{s}_{q2}}.$$

We output the proof $(\pi_1, \pi_2, \psi_1, \psi_2, s)$.

GSVerify($params_{GS}, (\pi_1, \pi_2, \psi_1, \psi_2, s)$. The verifier uses the commitments and values contained in the statement s to reconstruct $\tilde{c}_q$ and $\tilde{d}_q$ and outputs accept if

$$\prod_{q=1}^{Q} E(\tilde{c}_q, \tilde{d}_q) = \begin{pmatrix} 1 & 1 \\ 1 & t \end{pmatrix} E(u_1, \pi_1)E(u_2, \pi_2)E(\psi_1, v_1)E(\psi_2, v_2).$$

Groth and Sahai [GS07] prove that the scheme outlined above is complete, sound, and witness indistinguishable. Proofs consists of 4 group elements of G_1 and 4 group elements of G_2. Soundness comes from the fact that for the commitments correspond to the ElGamal ciphertexts, e.g. for G_1, Commit(x) corresponds to the ciphertext $(g^{r_1+s_1r_2}, (g^{z_1})^{r_1+s_1r_2}x)$ with public key g^{z_1} and secret key z_1. Consequently, we can extract x given z_1. In the case of GSExpCommit, we can use z_1 to extract b^{θ}. We get witness indistinguishability because if we choose $u_1, u_2 \in G_1 \times G_1$ and $v_1, v_2 \in G_2 \times G_2$, such that u_1 and v_1 are linear independent from u_2 and v_2 respectively, then commitments are perfectly hiding. By the SXDH assumption, we cannot distinguish the common reference string from such a simulated reference string.

L.2 Instantiation for Construction Based on Weak Boneh-Boyen Signatures

$\mathsf{Setup}(1^k)$**.** Runs $\mathsf{BilinearSetup}(1^k)$ to get $params = (p, G_1, G_2, G_T, e, g, h)$. Runs $\mathsf{GSSetup}(params)$ to get $params_1$ and $params_2$. Choose $u \leftarrow G$. Output $params = (p, params_1, params_2, G_T, e, u)$.

$\mathsf{ObtainSig}(params, pk, msg, comm, opening) \leftrightarrow \mathsf{IssueSig}(params, sk, comm)$**.** This does not depend on the GS proof system.

$\mathsf{Prove}(params, v, msg, \sigma)$**.** First, checks that σ is a valid signature on msg and terminates if it is not. Choose $s_{msg1}, s_{msg2}, r_{msg1}, r_{msg2}, r_{\sigma 1}, r_{\sigma 2}$. Compute

$$\begin{aligned} M_h &= \mathsf{GSExpCommit}(params_2, h, msg) = (1, h^{msg}) v_1^{s_{msg1}} v_2^{s_{msg2}} \\ M_u &= \mathsf{GSExpCommit}(params_1, u, msg) = (1, u^{msg}) u_1^{r_{msg1}} u_2^{r_{msg2}} \\ \Sigma &= \mathsf{Commit}(params_1, \sigma) = (1, \sigma) u_1^{r_{\sigma 1}} u_2^{r_{\sigma 2}}. \end{aligned}$$

We need to construct the proof

$$\mathsf{NIPK}\{((M_h : h^\alpha), (M_u : u^\beta), (\Sigma : x)) : \alpha = \beta \wedge e(x, vh^\alpha) = z)\}.$$

As described in Section 4.4, this is equivalent to the proof

$$\mathsf{NIPK}\{((M_h : H), (M_u : U), (\Sigma : x)) : e(u, H) \cdot e(U, h^{-1}) = 1 \wedge e(x, vH) = z)\}.$$

We need to do two pairing product equation proofs. First we prove $\mathsf{NIPK}\{((M_h : H), (M_u : U)) : e(u, H) \cdot e(U, h^{-1}) = 1\}$. $\mathsf{GSProve}$ computes "shadow" commitments to u as $(1, u)u_1^0 u_2^0 = (1, u)$ and h^{-1} as $(1, h^{-1})v_1^0 v_2^0 = (1, h^{-1})$. We choose $\{t_{i,j}\}_{i,j=1\ldots2} \leftarrow Z_p$ and compute

$$\begin{aligned} \pi_1 &= v_1^{t_{11}} v_2^{t_{12}} M_h^0 (1, h^{-1})^{r_{msg1}} & \psi_1 &= u_1^{-t_{11}} u_2^{-t_{21}} (1, u)^{s_{msg1}} (1, u^{msg})^0 \\ \pi_2 &= v_1^{t_{21}} v_2^{t_{22}} M_h^0 (1, h^{-1})^{r_{msg2}} & \psi_2 &= u_1^{-t_{12}} u_2^{-t_{22}} (1, u)^{s_{msg2}} (1, u^{msg})^0 \end{aligned}$$

Next we prove $\mathsf{NIPK}\{((M_h : H), (\Sigma : x)) : \wedge e(x, v \cdot H) \cdot e(x, pk) = z)\}$. $\mathsf{GSProve}$ computes a "shadow" commitment to vH as $vM_h = (1, vH)v_1^{s_{msg1}} v_2^{s_{msg2}}$. We choose $\{t'_{i,j}\}_{i,j=1\ldots2} \leftarrow Z_p$ and compute

$$\begin{aligned} \pi'_1 &= v_1^{t'_{11}} v_2^{t'_{12}} (vM_h)^{r_{\sigma 1}} & \psi'_1 &= u_1^{-t'_{11}} u_2^{-t'_{21}} (1, \sigma)^{s_{msg1}} \\ \pi'_2 &= v_1^{t'_{21}} v_2^{t'_{22}} (vM_h)^{r_{\sigma 2}} & \psi'_2 &= u_1^{-t'_{12}} u_2^{-t'_{22}} (1, \sigma)^{s_{msg2}} \end{aligned}$$

We output $comm = M_h$ and $\pi = (\pi_1, \pi_2, \psi_1, \psi_2, \pi'_1, \pi'_2, \psi'_1, \psi'_2, M_h, M_u, \Sigma)$. The rest of the statement is implicit in the construction and shared between prover and verifier.

$\mathsf{VerifyProof}(params, pk, comm, \pi = (\pi_1, \pi_2, \psi_1, \psi_2, \pi'_1, \pi'_2, \psi'_1, \psi'_2, M_h, M_u, \Sigma))$ outputs accept if $comm = M_h$ and

$$\begin{aligned} E\Big((1, u), M_h\Big) E\left(M_u, \begin{pmatrix} 1 \\ h^{-1} \end{pmatrix}\right) &= \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} E(u_1, \pi_1) E(u_2, \pi_2) E(\psi_1, v_1) E(\psi_2, v_2) \wedge \\ E\Big(\Sigma, \begin{pmatrix} 1 \\ v \end{pmatrix} M_h\Big) &= \begin{pmatrix} 1 & 1 \\ 1 & z \end{pmatrix} E(u_1, \pi'_1) E(u_2, \pi'_2) E(\psi'_1, v_1) E(\psi'_2, v_2), \end{aligned}$$

and reject otherwise. Note that multiplication is always elementwise and the pairing E is as defined above.[8]

[8] $\binom{a}{b} \cdot \binom{x}{y} = \binom{ax}{by}$, $E\left(\binom{a}{b}, (x, y)\right) = \begin{pmatrix} ax & ay \\ bx & by \end{pmatrix}$, and $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} w & x \\ y & z \end{pmatrix} = \begin{pmatrix} aw & bx \\ cy & dz \end{pmatrix}$.