Cryptanalyzing Bresson, et al.'s Spontaneous Anonymous Threshold Signature for Ad Hoc Groups and Patching via Updating Cramer, et al.'s Threshold Proof-of-Knowledge

Joseph K. Liu[1], Victor K. Wei[1], and Duncan S. Wong[2]

[1] Department of Information Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
{ksliu9,kwwei}@ie.cuhk.edu.hk
[2] Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
duncan@cityu.edu.hk

Abstract. We present an algebraic cryptanalysis of Bresson, et al.'s spontaneous anonymous threshold signature for ad hoc groups [6, 4]. The technique is to reduce a degenerate condition in Lagrange interpolation to an algebraically solvable high-density knapsack problem over $GF(2^\ell)$. We repair their protocol by revisiting and updating Cramer, et al.'s result on spontaneous anonymous threshold proof-of-knowledge (partial proof-of-knowledge). We generalize their proof by removing two assumptions, and reduce its security to a new candidate hard problem, PoK-Collision, in the random oracle model. To add to the urgency of our update, we present major versions of major PoK schemes that do not satisfy their special soundness assumption.

Feb 15, 2004.

1 Introduction

In an (n, t) signature scheme, there is a group of n members, and a pair of signature generation algorithm and signature verification algorithm such that

- t or more members in the group can jointly achieve a signature passing the verification;
- $t - 1$ or fewer members cannot generate a valid signature unless they can solve a hard problem.

In most papers on group signatures, even in most papers on group cryptography, the verification is based on proving knowledge of a *group secret*. There is usually a setup stage in which one of the following happens:

- (*Centralized group secret generation*) A group manager or a TTP (trusted third party) generates the group secret and distribute shares to all n group members [12, 13, 20, 14, 22].
- (*Distributed group secret generation*) All n members of the group jointly generate the group secret and distribute shares to all n members, often through multiple rounds of broadcasting or member-to-member secured communications [31, 18, 25, 23, 16, 2].
- (*Hybrid group secret generation*) A combination of the above two [9, 10, 7, 8].

Usually, members' personal key pairs are used to secure communications between the members, and between the members and the group manager, the TTP, the group clerk (combiner).

(*Spontaneous Anonymous Group Cryptography: Group cryptography without group secret.*) Recently, a new class of group signature protocols (and group cryptographic protocols) began to emerge [11, 32, 5, 1, 36, 27]. They satisfy the definition of (n,t) signatures, yet there is no group secret. (However, there is still a binary relation.) The verification is based on the public keys of the n members of the group.

There is no set up. To begin with, there is no need to setup for generating the group secret, or for secret-sharing the group secret. Furthermore, there is no setup for selecting members to form a group. Any t entities with published public keys can jointly select $n-t$ other entities with published public keys to form a group, and generate an (n,t)-signature without any participation from the $n-t$ conscripts (or diversion signers). In the case of 1-out-of-n signature, i.e. $t=1$, any single entity can spontaneously conscript $n-1$ diversion signers and complete the signature single-handedly.

These schemes are anonymous in the sense that they are signer-indistinguishable. The verifier cannot tell the actual signers (insiders) from the diversion signers. To be more specific, almost all examples of this class of schemes we list above have

- proven unconditional (information-theoretic) anonymity;
- proven unconditionally irrevocable anonymity;
- proven unconditionally exculpable anonymity.

Even if all secret keys of all n members are revealed, along with transcripts of all communications between insiders (actual signers) and outsiders (all other parties), the anonymity remains information-theoretically secure. This is just about the *maximum privacy* one can possibly imagine.

The combination of spontaneity and maximum privacy, makes spontaneous anonymous group (SAG) cryptography uniquely suited to many applications. We only name a few here.

Whistle Blowing (or *How to Leak a Secret* [32]) Deepthroat leaks confidential information to Washington Post reporters. He wants information-theoretic, irrevocable, exculpable anonymity. Nothing less than that will do. The editor wants proof that the information is indeed from a White House Staff before publishing it. So Deepthroat generates an SAG signature using public keys of the entire

White House staff. Editor is satisfied and publishes the article. Deepthroat's identify is protected forever. The rest, as they say, is history.

Ad Hoc Group Cryptography. In an ad hoc group, members join and leave frequently. Furthermore, there is minimal infrastructural support for complicated multi-round multi-party protocols. It has been observed by many authors that the spontaneity in SAG cryptography makes it perfectly suited to cryptographic applications in ad hoc groups [5, 6, 4, 3]. There can also be applications to mobile communications and sensor networks.

The first SAG cryptography result was in Cramer, et al. [11]. The paper by Rivest, et al. [32] brought envigored interests to the field. Currently the main approaches to constructing SAG signatures include: the ring structure where the computations can be block-diagramed as a circular ring [32, 1], the Boolean structure [5, 6, 4], the secret-sharing structure [11, 6, 4, 27], and the bilinear structure [3]. There are also some ring-structure signatures that become fully spontaneous only after additional assumptions [37].

1.1 Our Contributions

We present an algebraic cryptanalysis of a threshold SAG (spontaneous anonymous group) signature scheme for ad hoc group applications by Bresson, et al. [6, 4]. A degenerate condition in Lagrange polynomial interpolation is used to reduce a signature forgery problem to a high-density knapsack problem over $GF(2^\ell)$ which is then solved algebraically. It is a new vulnerability in SAG cryptography which does not have known counterparts in group cryptography with group secret. We think all future research in group cryptography without group secrets should pay attention to this new type of vulnerabilities.

We also present two patches. The first (naive) patch remains exposed to potential cryptanalysis by research tools from high-density knapsack problems over a large prime field. This provides further evidence that our new cryptanalysis poses a wider threat to future SAG protocols in general. The second patch is proven secure by reducing it to a new candidate hard problem: the PoK(Proof-of-Knowledge)-Collision Problem.

The PoK-Collision Problem: Given a 3-move PoK scheme, generate a pair of transcripts $(commit, challenge, response)$ and $(commit', challenge', response')$ satisfying $commit = commit'$.

The process of proving the second patch led us to revisit Cramer, et al. [11]. The paper contained pioneering results on proof-of-knowledge, authentication, and signature in threshold SAG cryptography. We solve the two open problems posed in the concluding section of that paper by generalizing its proof in removing two assumptions: that each component PoK (proof-of-knowledge) protocol satisfies *special soundness*, and public flip coins are used. Essentially, we used their protocol unaltered, but updated its proof by reducing its security to the hardness of the PoK-Collision Problem, in the random oracle model.

We also present, and prove, major versions of major PoK schemes that our new result above applies to but the original Cramer, et al.'s result does not apply to. The Feige-Fiat-Shamir PoK is included. They justify the need to revisit and update Cramer, et al.[11].

Organization. In Sec. 2, a threshold SAG signature scheme is formalized with a brief description of the security model. Current SAG signature schemes are reviewed. In Sec. 3, an algebraic cryptanalysis technique is described and the vulnerability of the scheme of Bresson, et al. is explained. A would-be patch of their scheme is discussed. In Sec. 4, a threshold proof of knowledge protocol due to Cramer, et al. is generalized and converted into a threshold SAG signature scheme. The scheme is shown to be secure against adaptive adversary under the random oracle model. The paper is concluded in Sec. 5.

2 Related Work

A t-out-of-n threshold SAG (Spontaneous Anonymous Group) signature scheme is a triple ($\mathcal{G}$, $\mathcal{S}_{t,n}$, $\mathcal{V}_{t,n}$) where

- $(\hat{s}, P) \leftarrow \mathcal{G}(1^k)$ is a probabilistic algorithm which takes security parameter k and outputs private key $\hat{s}$ and public key P.
- $\sigma \leftarrow \mathcal{S}_{t,n}(1^k, \hat{S}, L, m)$ is a probabilistic algorithm which takes as inputs security parameter k, a set of t private keys $\hat{S}$, a set of n public keys L including the ones that correspond to the private keys in $\hat{S}$ and message m, produces a signature σ.
- $1/0 \leftarrow \mathcal{V}_{t,n}(1^k, L, m, \sigma)$ is an algorithm which accepts as inputs security parameter k, a set of n public keys L, a message m and a signature σ, returns 1 or 0 for accept or reject, respectively. We require that

  $$\mathcal{V}_{t,n}(1^k, L, m, \mathcal{S}_{t,n}(1^k, \hat{S}, L, m)) = 1$$

  for any message m, any set $\hat{S}$ of t private keys and any set L of n public keys in which the public keys corresponding to all the private keys of $\hat{S}$ are included. For each key in L indexed by $i \in \{1, \cdots, n\}$, we assume that the key pair $(\hat{s}_i, P_i)$ is generated by $\mathcal{G}$.

For simplicity, we omit the security parameter as an input of $\mathcal{S}_{t,n}$ and $\mathcal{V}_{t,n}$ in the rest of the paper when it becomes clear.

A t-out-of-n threshold SAG signature scheme must satisfy the usual correctness and unforgeability properties. If all the involving parties are honest and the scheme is properly executed, the generated threshold SAG signature should be accepted as valid with respect to the set of public keys L with overwhelming probability; and it must be infeasible for anyone, except with negligible probability, to generate a valid threshold SAG signature with respect to L if he does not have the threshold-of-t private keys corresponding to keys of L. We skip the formal description of the security model due to the page limitation.

A threshold SAG signature should also be signer anonymous, in the sense that no one even with unbounded resources can guess one of the t authors of a threshold SAG signature with respect to L with probability non-negligibly greater than t/n.

Spontaneity is the crucial feature which distinguishes an SAG signature scheme from a conventional group signature scheme or a threshold signature scheme. It is required that any set of t entities can form an SAG (i.e. an ad hoc group) with respect to L by including their own public keys and the public keys of some other $n-t$ diversion entities into L. The threshold SAG signature can be generated using their private keys and the public keys of those diversion entities. The only assumption is that each entity in the system is already associated with the public key of some standard signature scheme (via a PKI directory or certificate).

The concept of threshold SAG signature scheme first appeared in [11] in Crypto 94 as a threshold proof of knowledge protocol. When the scheme is turned into a signature scheme using standard methods, it convinces a verifier that a document is signed by one (or t, in the case of threshold setting) of the n possible signers without allowing the verifier to identify the actual signer(s).

In [32], a ring-based SAG signature scheme was proposed and its security was shown under the ideal cipher model. Several other ring-based SAG signature schemes were later proposed [36, 15]. In [5, 6, 4], Bresson, et al. proposed two threshold SAG signature schemes. One scheme is based on boolean construction and the other scheme uses the polynomial interpolation technique. We show in Sec. 3 that their polynomial-based scheme is vulnerable to an attack based on some algebraic methods when $n-t$ is large where n is the group size and t is the threshold value. Other polynomial construction based (threshold) SAG signature schemes can be found in [36, 19].

In [3], an aggregate signature scheme was proposed. It uses the special property of a bilinear group (the group where Decisional Diffie-Hellman problem is easy while Computational Diffie-Hellman problem is hard) to construct a 1-out-of-n SAG signature scheme which enjoys short signature and aggregation property.

Additional properties of (threshold) SAG signature schemes can also be found in literature. In [1, 27], several schemes have been proposed which have the separability property, that is, allowing the mixture of keys with different key types in one SAG. In [37], an ID-based SAG signature scheme was proposed. It is considered to be a type of SAG signature scheme provided that the TTP is completely trustworthy. Hence we consider the scheme to be a *partial* SAG signature scheme.

3 Algebraic Cryptanalysis of Bresson, et al.'s SAG Threshold Signature

We review the captioned scheme and present a polynomial-time algebraic cryptanalysis. The review first.

Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a hash function where ℓ is the security parameter. For each user i, $1 \le i \le n$, let P_i be the public key. Let F and E be a family of trapdoor permutations and a family of symmetric encryption functions, respectively. Each function in F and E is defined over $\{0,1\}^\ell$. Without loss of generality, suppose the actual signers are indexed by $1, \cdots, t$. Below is the signature generation algorithm for message m.

1. Compute $c_0 \leftarrow H(P_1, \cdots, P_n)$. For $i = t+1, \cdots, n$, do $x_i \overset{R}{\leftarrow} \{0,1\}^\ell$ and $y_i \leftarrow F_{P_i}(x_i)$
2. Interpolate a polynomial f over GF(2^ℓ) such that $\deg(f) = n-t$, $f(0) = c_0$, and $f(i) = E_{H(m,i)}(y_i)$, for $i = t+1, \cdots, n$.
3. For $i = 1, \cdots, t$, do $x_i = F_{P_i}^{-1}(E_{H(m,i)}^{-1}(f(i)))$.
4. Output the signature tuple $\sigma = (m, P_1, \cdots, P_n, x_1, \cdots, x_n, f)$.

The verification is to check if $f(0) = H(P_1, \cdots, P_n)$ and $f(i) = E_{H(m,i)}(F_{P_i}(x_i))$ for $i = 1, \cdots, n$.

We now describe a polynomial-time forgery algorithm. There are three steps to understand it.

1. We derive a condition which degenerates the polynomial interpolated from $n-t+2$ evaluations to degree $n-t$.
2. We convert that condition to a high-density knapsack problem over $GF(2^\ell)$.
3. We algebraically solve the high-density knapsack problem over $GF(2^\ell)$.

The subset sum (knapsack) problem [35]. Given integers B and $A_1, \cdots, A_{n'}$, a *Decisional Subset Sum Problem* is to determine if there exists binary values $b_1, \cdots, b_{n'} \in \{0,1\}$ such that $B = \sum_{i=1}^{n'} b_i A_i$. $(B, A_1, \cdots, A_{n'})$ is called a cargo vector. A *Computational Subset Sum Problem* over the same cargo vector is defined as computing the binary values b_1, $\cdots$, b_n when the decisional knapsack problem returns positive.

Lemma 1 (Degenerate condition in polynomial interpolation). *Given c_0 and c_i, $i = t, \cdots, n$. A polynomial f of degree $n-t$ or less and $f(0) = c_0$, $f(i) = c_i$, $i = t, \cdots, n$, exists if and only if*

$$[f(0)\ f(t)\ f(t+1)\ \cdots\ f(n)] \cdot [v_0\ v_t\ v_{t+1}\ \cdots\ v_n] = 0$$

for a specific vector $\mathbf{v} = [v_0\ v_t\ v_{t+1}\ \cdots\ v_n]$ whose is the dual to the vector space formed by the following $n-t$ row vectors

$$\mathbf{e}_i = [0^i\ t^i\ (t+1)^i\ \cdots\ n^i],\ 0 \le i \le n-t$$

Proof. Let $f(x) = \sum_{i=0}^{n-t} f_i x^i$ denote a polynomial of degree $n-t$ or less. Then

$$\mathbf{f} := [f(0)\ f(t)\ f(t+1)\ \cdots\ f(n)] = \sum_{i=0}^{n-t} f_i \mathbf{e}_i$$

and consequently $\mathbf{f} \cdot \mathbf{v} = 0$. Conversely if a vector $\mathbf{f}$ is in the dual space, i.e. $\mathbf{f} \cdot \mathbf{v} = 0$, then it can be expressed as a linear combination of the base vectors as follows

$$\mathbf{f} = \sum_{i=0}^{n-t} f_i \mathbf{e}_i$$

□

3.1 A Forgery Algorithm

Without loss of generality, suppose the forger $\mathcal{A}$ knows the private keys of $P_1, \cdots, P_{t-1}$.

1. Compute $c_0 \leftarrow H(P_1, \cdots, P_n)$. For each i, $t \le i \le n$, randomly generate $x_{i,0}$ and $x_{i,1}$. Compute $y_{i,b} = F_{P_i}(x_{i,b})$ and $c_{i,b} = E_{H(m,i)}(y_{i,b})$ for $b \in \{0,1\}$.
2. Solve the subset sum problem with
$$(B, A_t, \cdots, A_n) = (-c_0 v_0 - \sum_{i=t}^{n} c_{i,0} v_i, (c_{t,1} - c_{t,0}) v_t, \cdots, (c_{n,1} - c_{n,0}) v_n) \quad (1)$$
to obtain a solution $(b_t, \cdots, b_n) \in \{0,1\}^{n-t+1}$ satisfying
$$\begin{aligned} -c_0 v_0 - \sum_{i=t}^{n} c_{i,0} v_i &= \sum_{i=t}^{n} b_i (c_{i,1} - c_{i,0}) v_i \\ &= \sum_{i=t}^{n} (c_i - c_{i,0}) v_i \end{aligned}$$
where $c_i = c_{i,b_i}$ for each i, $t \le i \le n$. Then we have
$$c_0 v_0 + \sum_{i=t}^{n} c_i v_i = 0$$
i.e. $\mathbf{c} \cdot \mathbf{v} = 0$.
3. Interpolate a polynomial f of degree n–t satisfying $f(0) = c_0$ and $f(i) = c_i = c_{i,b_i}$ for $t \le i \le n$. By the above equality and Lemma 1, such a polynomial exists.
4. For each i, $1 \le i \le t-1$, $x_i \leftarrow F_{P_i}^{-1}(E_{H(m,i)}^{-1}(f(i)))$.
5. Output the signature $(m, P_1, \cdots, P_n, x_1, \cdots, x_n, f)$.

The density of the knapsack problem is roughly
$$d < \frac{n-t}{\text{avg.} \log_2(|c_{i,1} - c_{i,0}| v_i)} \approx \frac{n-t}{\ell - 1} \approx \frac{n-t}{\ell}$$

When $n-t > \ell$, the chance of having the cargo vector in (1) be a valid subset sum problem is non-negligible. Our forgery algorithm above is required to solve the computational subset sum problem: Given $B, A_t, \cdots, A_n \in GF(2^\ell)$, find $(b_t, \cdots, b_n) \in \{0,1\}^{n-t+1}$ satisfying $\sum_{i=t}^{n} b_i A_i = B$ in GF(2^ℓ).

The above is positive and polynomial-time computable when $n-t > \ell$. Let α be a primitive element of GF(2^ℓ). Then $\{\alpha^{2^i} : 0 \le i \le \ell - 1\}$ form a (standard) basis of GF(2^ℓ). For each element $\beta \in$ GF(2^ℓ), there exists a unique ℓ-tuple $(\beta_0, \cdots, \beta_{\ell-1} \in \{0,1\}^\ell$ such that $\beta = \sum_{i=0}^{\ell-1} \beta_i \alpha^{2^i}$. Therefore the elements of GF(2^ℓ) form a vector space of dimension ℓ over GF(2). In the forgery algorithm, if $n-t > \ell$, then there exists a linear dependence relationship can be computed in polynomial time to complete the signature generation.

From (1), let

$$B = \sum_{i=0}^{\ell-1} \beta_i \alpha^{2^i}$$

where α is a primitive element of GF(2^ℓ), $\beta_i \in \{0,1\}$. $\{\alpha^{2^i} : 0 \leq i \leq \ell-1\}$ forms a (standard) basis of GF(2^ℓ). For each element $B \in$ GF(2^ℓ), there exists a unique ℓ-tuple $(\beta_0, \cdots, \beta_{\ell-1}) \in \{0,1\}^\ell$ such that $B = \sum_{i=0}^{\ell-1} \beta_i \alpha^{2^i}$.

Let

$$A_i = \sum_{j=0}^{\ell-1} \gamma_{i,j} \alpha^{2^j}, \quad t \leq i \leq n$$

where $\gamma_{i,j} \in \{0,1\}$.

Find $(b_t, \cdots, b_n) \in \{0,1\}^{n-t+1}$ such that

$$\begin{bmatrix} \beta_0 \\ \vdots \\ \beta_{\ell-1} \end{bmatrix} = \begin{bmatrix} \gamma_{t,0} & \gamma_{t+1,0} & \cdots & \gamma_{n,0} \\ \vdots & \vdots & & \\ \gamma_{t,\ell-1} & \gamma_{t+1,\ell-1} & \cdots & \gamma_{n,\ell-1} \end{bmatrix} \begin{bmatrix} b_t \\ \\ \vdots \\ \\ b_n \end{bmatrix}$$

We have ℓ equations but $n-t+1$ unknowns (Note: $n-t+1 \geq \ell$). Hence a sequence of $(b_t, \cdots, b_n)$ can be chosen to complete the signature generation.

In the special case when $t = 1$, that is, generating a 1-out-of-n SAG signature, the attacking technique works as long as n is comparable to ℓ (if not greater than).

To the best of our study of the proof of Theorem 1 in [6], we agree to the probability that there exists two $E_{H(m,i)}$-query for which the answer equals $f(i)$ is at most $q_E/2^\ell$ but we disagree with the argument that there are at least t indices for which $\mathcal{A}$ made an E^{-1}-query on $f^*(i)$.

Remark: One observation is that the computation of c_0 may not be chosen carefully in the BSS algorithm. The attack works because of the irrelevancy between the computation of c_0 and the choices of $c_t, \cdots, c_n$.

In the following, we evaluate a would-be patch of BSS scheme. In this patch, f is over GF(p) where the prime $p \approx 2^\ell$. We argue that this would-be patch may also be crackable when $n-t$ is large. In particular, the important case when $t = 1$ may also be at risk in this scenario.

3.2 A naive patch that may not work

Cryptanalysis above shows that the original BSS scheme is susceptible to an attack which is provably polynomial time for f being over GF(2^ℓ). We now evaluate a would-be patch which turns the polynomial f from over GF(2^ℓ) to over GF(p) where the prime $p \approx 2^\ell$. We assume other needed modifications are also feasible without going into details.

The vulnerability of this modified BSS scheme lies with the high-density knapsack problem. The *density* of the subset sum problem is

$$d \approx \frac{n-t}{\ell}$$

The problem is a *low-density knapsack problem* if $d < 1$. Low-density knapsack problems with $d < 2/n'$ are provably polynomial time [28]. Main technique is the LLL (Lenstra-Lenstra-Lovasz) lattice reduction algorithm [24]. Many low-density knapsack problems with $2/n' < d < 1$ are almost provably polynomial time, and/or considered insecure due to empirical results. Knapsack problems with large high densities are often crackable [21]. However, the general knapsack problem is proven NP-complete, and many researchers continue to believe, with the right balance of design parameters around a knapsack problem with density near 1, a secure asymmetric crypto-system can be constructed and proven NP-complete to cryptanalyze [21].

If $n - t = (1 - \epsilon_1)\ell$, where the constant $\epsilon_1 > 0$, then the probability of the existence of a polynomial f in the forgery algorithm is roughly $2^{-\epsilon_1 \ell}$, negligible. The modified BSS algorithm may be secure in this case. So assume $n - t = (1+\epsilon_2)\ell$, where the constant $\epsilon_2 > 0$. When ϵ_2 is large, the high-density knapsack problem tends to be insecure, and so does the modified BSS algorithm. Note that the important case $t = 1$ may be at risk in this scenario.

4 Generalizing the Proof of Cramer, et al.'s Threshold Proof of Knowledge

We present the captioned generalization. In fact, we only generalize its proof by removing two requirements: that component PoK's need to satisfy only ordinary soundness not special soundness, and that random oracle replaces public coins. We do not alter their protocol. For simplicity, we assume a polynomial-based secret sharing, instead of the monotone access structure contained in the original paper, is used. We adopt their notations, and their main protocol in Section 4.

4.1 Security Model and Results.

Entities: Dealer $\mathcal{D}$, Simulator $\mathcal{S}$, Forger $\mathcal{F}$.

The Game, assume the component PoK protocol has been given:

1. $\mathcal{D}$ generates n, t, and n public keys, gives them to $\mathcal{S}$ and asks for t PoK-Collision pairs corresponding to t of the n public keys.
2. $\mathcal{S}$ constructs a set of n public keys, and asks $\mathcal{F}$ to produce an (n, t) threshold PoK.
3. $\mathcal{F}$ computes and makes queries to signing oracle $\mathcal{SO}$ and random oracle H, both simulated by $\mathcal{S}$, and then delivers a valid answer to $\mathcal{S}$.
4. $\mathcal{S}$ computes and solves $\mathcal{D}$'s hard problem instantiations.

Further model details:
The signing oracle Model. $\mathcal{F}$ can query $\mathcal{SO}$ with any set of public keys. But n and t are fixed.

The adaptive adversary model. We use just about the most challenging adaptive adversary model imaginable. $\mathcal{S}$ does not know any secret key in the hard problem instantiation received from $\mathcal{D}$. $\mathcal{F}$ can adaptively corrupt any $t-1$ secret keys in the problem it receives from $\mathcal{S}$, with erasure-freeness, active adversary, and all other advantages. $\mathcal{SO}$ does not know any secret key in queries to it.

Remark: It is interesting to observer that security against very strong adaptive adversary models can be routinely proved for group cryptography without group secret, while even much weaker adaptive adversary models pose daunting challenges in group cryptography with group secrets.

Definition 1 ((n,t)-PoK). *Given a three-move PoK protocol P, public keys w_i, $1 \le i \le n$, the triple of n-vectors $(\mathbf{m}_1, \mathbf{c}, \mathbf{m}_2)$ is an (n,t)-PoK if there exists a polynomial f of degree no more than $n-t$ such that $f(i) = c^{(i)}$, $1 \le i \le n$, and $f(0) = H(m_1^{(1)}, \cdots, m_1^{(n)})$, and each conversation $(m_1^{(n)}, c^{(n)}, m_2^{(n)})$ is valid for w_i, $1 \le i \le n$, where $\mathbf{m}_1 = (m_1^{(1)}, \cdots, m_1^{(n)})$, $\mathbf{c} = (c^{(1)}, \cdots, c^{(n)})$, where $\mathbf{m}_2 = (m_2^{(1)}, \cdots, m_2^{(n)})$.*

Definition 2 (Existential Forger). *A probabilistic Turing machine $\mathcal{F}$ is an existential forger if it can produce (n,t)-PoK with non-negligible probability when given n public keys by $\mathcal{D}$.*

Definition 3 (PoK-Collision). *Given P which is a three-move honest verifier zero-knowledge proof of knowledge for relation $\mathcal{R}$ satisfying soundness, produce two conversations (m_1, c, m_2) and (m_1, c', m_2'), $c \neq c'$, valid in P.*

Theorem 1. *There exists a PPT Turing Machine for generating (n,t)-PoK spontaneously which is existentially unforgeable by adaptive chosen public key adersaries in the random oracle model, provided PoK-Collision is hard.*

4.2 Proof

Our proof agenda is as follows:

1. Specify a pair of (n,t)-PoK generation and verification algorithm. Confirm completeness, soundness, spontaneity.
2. Specify $\mathcal{SO}$ simulation.
3. Specify witness extraction, and the condition when this succeeds.
4. Specify the entire Simulator $\mathcal{S}$, its complexity and success probability.

Protocol: VERIFY. It is straightforward to verify the definition.

Protocol: Generate (n,t)-PoK. Denote the given public keys as $w_1, \cdots, w_n$. Denote the given secret keys as $\{x_i : i \in I\}$ where $I \subset \{1, \cdots, n\}$, $|I| = t$. Denote the given component PoK as P. The generation algorithm:

1. For each $i \notin I$, simulates P as described in [11] to produce a valid conversation $(m_1^{(i)}, c^{(i)}, m_2^{(i)})$.
2. For each $i \in I$, randomly picks $m_1^{(i)}$.
3. Compute $c_0 = H(\mathbf{m}_1)$.
4. Interpolate a polynomial f satisfying $f(i) = c_i$ for each $i \notin I$ and $f(0) = c_0$. Note the degree of f is no more than $n - t$.
5. For each $i \in I$, let $c^{(i)} = f(i)$ and use the given secret key x_i to compute $m_2^{(i)}$ and produce a valid conversation $(m_1^{(i)}, c^{(i)}, m_2^{(i)})$.
6. Output the (n,t)-PoK $(\mathbf{m}_1, \mathbf{c}, \mathbf{m}_2)$.

Completeness, soundness, spontaneity are straightforward. We do not treat robustness in this paper.

Simulating $\mathcal{SO}$. Given public keys w_1, $\cdots$, w_n, generate (n,t)-PoK.

1. Randomly pick $c^{(i)}$, $0 \le i \le n$, subject to the condition that the $n+$ evaluations $f(i) = c^{(i)}$, $0 \le i \le n$, interpolate a polynomial f with degree no more than $n - t$.
2. For each i, $1 \le i \le n$, simulate P as described in [11] to produce a valid conversation $(m_1^{(i)}, c^{(i)}, m_2^{(i)})$.
3. Back patch the random oracle to $H(\mathbf{m}_1) = c_0$.
4. Output the (n,t)-PoK $(\mathbf{m}_1, \mathbf{c}, \mathbf{m}_2)$.

Witness Extraction $\mathcal{WE}$. $\mathcal{S}$ extracts t witnesses for t of the n hard problem instantiations via one rewind, at the ℓ-th H query. Denote the ℓ-th query as $H(\mathbf{m}_{1,\ell})$ and denote its outputs in two first forks as c_0 and $\hat{c}_0$ respectively. Denote the results delivered by $\mathcal{F}$ in the two forks as $(\mathbf{m}_1, \mathbf{c}, \mathbf{m}_2)$ and $(\hat{\mathbf{m}}_\mathbf{1}, \hat{\mathbf{c}}, \hat{\mathbf{m}}_\mathbf{2})$ respectively.

If $\mathbf{m}_1 = \mathbf{m}'_1 = \mathbf{m}_{1,\ell}$, then $\mathcal{S}$ achieves desired witness extraction: By properties of (n,t) secret-sharing, there exists $I \subset \{1, \cdots, n\}$, $|I| = t$, such that $c^{(i)} \ne \hat{c}^{(i)}$, and $(m_1^{(i)}, c^{(i)}, m_2^{(i)})$ and $(\hat{m}_1^{(i)}, \hat{c}^{(i)}, \hat{m}_2^{(i)})$ is a PoK-Collision pair.

The sequencing, complexity, and probability of $\mathcal{S}$. We use the classification proof technique. Assume $\mathcal{F}$ is a (T, ϵ)-forger, which makes q_H queries to the random oracle and makes q_S queries to the signing oracle. For ℓ, $1 \le q_H + q_S$, let p_ℓ denote the probability that $\mathcal{F}$ succeeds in forgery and the ℓ-th random oracle query, counting in those made by $\mathcal{SO}$, is the H-query used in verifying $\mathcal{F}$'s result.

By the lunchtime attack argument, it is of negligible probability that $\mathcal{F}$ does not make the query $H(\mathbf{m}_1)$ where $\mathbf{m}_1$ is contained in the result $\mathcal{F}$ eventually delivers to $\mathcal{S}$. Therefore, there exists ℓ, such that $p_\ell \ge \epsilon/(q_H + q_S)$. If $\mathcal{S}$ rewinds at the ℓ-th query, then $\mathcal{F}$ succeeds in the second query also with probability $\epsilon/(q_H + q_S)$ according to the RoS Lemma [26]. Alternatively, the forking lemma of the heavy-row lemma can be used, albeit with inferior simulation efficiency.

For each $1 \le \ell \le q_H + q$, $\mathcal{S}$ makes a simulation run rewinding at the ℓ-th query. Then $\mathcal{S}$ is a $(2T(q_H + q_S), \epsilon^2(q_H + q_S)^{-2})$ Turing Machine to solve t of n of $\mathcal{D}$'s hard problem instantiations. □

.

4.3 Discussions

In the reduction proof of [11], the security of the protocol is actually reduced to PoK-Collision. Then by the assumption of another property of the component PoK, namely the *special soundness property* defined below, a witness $\hat{s}$ of a problem instance in $\{P_1, \cdots, P_n\}$ can be obtained in polynomial time with non-negligible probability.

Definition 4 (Special Soundness Property). *Given a problem instance P, let $c_1, c_2 \in_R \{0,1\}^k$ for some parameter k. For any prover $\tilde{P}$, given any two conversations between $\tilde{P}$ and $\mathcal{V}$, (m', c_1, m_1'') and (m', c_2, m_2'') with respect to P, where $c_1 \neq c_2$, an element of $\hat{s}(P)$ can be computed in polynomial time.*

Hence the special soundness property implies that PoK-Collision is equivalent to finding a witness of a problem instance P.

Why is ordinary soundness important? Cramer, et al. stated that [11] *"all known proofs of knowledge have [special soundness] property, or at least a variant where computation of the witness follows from some small number of correct answers."* The Guillou-Quisquater's RSA root protocol [17], Parallel version of Ohta-Okamoto's identification protocol [29], Schnorr's identification protocol [34], and an extension due to Okamoto [30] satisfy special soundness.

However, there are major versions of major PoKs that do not satisfy special soundness, e.g. Feige-Fiat-Shamir, and new PoK's that arise from SAG cryptography [27]. Therefore, there is added importance to update the Cramer, et al. result.

Now we give details on Feige-Fiat-Shamir PoK's lack of special soundness. First we review the version in [33]. Let n be the product of two large primes. The public key of the Prover (Peggy) is $\mathbf{v} = (v_1, \cdots, v_k)$ where each v_i is a QR in Z_n, and her secret key is $\mathbf{s} = (s_1, \cdots, s_k)$ where each s_i is the smallest square root of v_i in Z_n. It is a t-round protocol, each round consists of three moves. In the τ-th round, $1 \leq \tau \leq t$, the three moves are $(x_\tau, \mathbf{e}_\tau, y_\tau)$ where

1. Prover picks $r_\tau \in \{1, \cdots, n-1\}$, sends $x_\tau = r_\tau^2 \bmod n$ to Verifier.
2. Verifier picks $\mathbf{b}_\tau = (b_{\tau,1}, \cdots, b_{\tau,k}) \in \{0,1\}^k$.
3. Prover sends back $y_\tau = r_\tau \prod_{j=1}^k s_{\tau,j}^{b_{\tau,j}} \bmod n$.

The verification at the end of the three moves is that $x_\tau = y_\tau^2 \prod_{j=1}^k v_j^{b_{\tau,j}}$. We convert it to a one-round three-move protocol in the natural way: vectorization. Let the three-move conversation be denoted $(\mathbf{x}, \mathbf{b}, \mathbf{y})$ where $\mathbf{x} = (x_1, \cdots, x_t)$, $\mathbf{b} = (\mathbf{b}_1, \cdots, \mathbf{b}_t)$, $\mathbf{y} = (y_1, \cdots, y_t)$.

To prove Feige-Fiat-Shamir does not have special soundness, we exhibit a qualified pair of conversations that does not allow polynomial-time extraction of Prover's secret $\mathbf{s}$. The pair consists of $(\mathbf{x}, \mathbf{b}, \mathbf{y})$ and $(\mathbf{x}, \mathbf{b}', \mathbf{y}')$ where

$$\{\mathbf{b}_\tau \oplus \mathbf{b}'_\tau : 1 \leq \tau \leq t\} = 0^{k - \log_2 t} || \{0,1\}^{\log_2 t}.$$

Then only the tail $\log_2 t$ bits of Prover's secret $\mathbf{s}$ can be extracted, which do not constitute a witness.

5 Conclusions

We present an algebraic cryptanalysis of a threshold SAG signature scheme by Bresson, et al. [5,6,4]. Their design took a chance away from safe overengineering by not including enough parameters in their hash inputs. There are reasons to suspect that these renowned researchers knew the risk but still braved the research challenge in using minimal hash inputs for the noble purpose of exploring theoretical boundaries.

It is found that their scheme is insecure when $n - t$ is large. This includes the important case $t = 1$. The new field of SAG cryptography is intrigued with hitherto unknown vulnerabilities, that researchers should be on the lookout for.

We also generalize the proof of Cramer, et al.'s [11] threshold PoK protocol to remove two assumptions. We use their protocols without alteration. In retrospect, our generalization can be viewed as merely an update of their proof using techniques developed after the publication of their paper.

References

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *Proc. ASIACRYPT 2002*, pages 415–432. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2501.
2. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Proc. CRYPTO 97*, pages 425–439. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1294.
3. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc. EUROCRYPT 2003*, pages 416–432. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2656.
4. E. Bresson. *Cryptographic Protocols for Group Anonymity and Authentication.* Ph.D. Thesis, 2002.
5. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Proc. CRYPTO 2002*, pages 465–480. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2442.
6. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. full version. http://www.di.ens.fr/~bresson, 2002.
7. J. Camenisch. Efficient and generalized group signatures. In *Proc. EUROCRYPT 97*, pages 465–479. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1233.
8. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Proc. CRYPTO 97*, pages 410–424. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1294.
9. D. Chaum and E. van Heyst. Group signatures. In *Proc. EUROCRYPT 91*, pages 257–265. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 547.
10. L. Chen and T. Pedersen. New group signature schemes. In *Proc. EUROCRYPT 94*, pages 171–181. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 950.
11. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. CRYPTO 94*, pages 174–187. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 839.

12. Y. Desmedt. Society and group oriented cryptography: A new concept. In *Proc. CRYPTO 87*, pages 120–127. Springer-Verlag, 1987. Lecture Notes in Computer Science No. 293.
13. Y. Desmedt and Y. Frankel. Threshold cryptosystem. In *Proc. CRYPTO 89*, pages 307–315. Springer-Verlag, 1989. Lecture Notes in Computer Science No. 435.
14. Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In *Proc. CRYPTO 91*, pages 457–469. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 576.
15. C. Gao, Z. Yao, and L. Li. A ring signature scheme based on the nyberg-rueppel signature scheme. In *Applied Cryptography and Network Security (ACNS 2003)*, pages 169–175. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2846.
16. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of rsa functions. In *Proc. CRYPTO 96*, pages 157–172. Springer-Verlag, 1996. Lecture Notes in Computer Science No. 1109.
17. L. Guillou and J. J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Proc. EUROCRYPT 88*, pages 123–128. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 330.
18. L. Harn. Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *IEE Proc. Computers and Digital Techniques*, 141(5):307–313, 1994.
19. J. Herranz and G. Saez. Forking lemmas for ring signature schemes. In *Progress in Cryptology - INDOCRYPT 2003*, pages 266–279. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2904.
20. T. Hwang. Cryptosystem for group oriented cryptography. In *Proc. EUROCRYPT 90*, pages 352–360. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 473.
21. M. K. Lai. Knapsack cryptosystems: The past and the future. Available online at http://www.ics.uci.edu/~mingl/knapsack.html, 2001.
22. C. Laih and L. Harn. Generalized threshold cryptosystems. In *Proc. ASIACRYPT 91*, pages 159–166. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 739.
23. S. K. Langford. Threshold DSS signatures without a trusted party. In *Proc. CRYPTO 95*, pages 397–409. Springer-Verlag, 1995. Lecture Notes in Computer Science No. 963.
24. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Ann.*, 261:513–534, 1982.
25. C. Li, M. Hwang, and N. Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In *Proc. EUROCRYPT 94*, pages 194–204. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 950.
26. J. K. Liu, V. K. Wei, and D. S. Wong. Linkable and culpable ring signatures. Technical Report 027, IACR ePrint, 2004.
27. J. K. Liu, V. K. Wei, and D. S. Wong. A separable threshold ring signature scheme. In *ICISC 2003*, pages 7–22. Springer-Verlag, 2004. To appear in Lecture Notes in Computer Science series.
28. P. Q. Nguyen and J. Stern. The two faces of lattices in cryptography. In *Proc. of Cryptography and Lattices Conference.* Springer-Verlag, 2001. Lecture Notes in Computer Science No. 2146.

29. K. Ohta and T. Okamoto. A modification of the fiat-shamir scheme. In *Proc. CRYPTO 88*, pages 232–243. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 403.
30. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Proc. CRYPTO 92*, pages 31–53. Springer-Verlag, 1993. Lecture Notes in Computer Science No. 740.
31. T. Pedersen. A threshold cryptosystem without a trusted party. In *Proc. EUROCRYPT 91*, pages 522–526. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 547.
32. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proc. ASIACRYPT 2001*, pages 552–565. Springer-Verlag, 2001. Lecture Notes in Computer Science No. 2248.
33. Bruce Schneier. *Appllied Cryptography.* Wiley, 1996.
34. C. P. Schnorr. Efficient identification and signatures for smart cards. In *Proc. CRYPTO 89*, pages 239–252. Springer, 1990. Lecture Notes in Computer Science No. 435.
35. Douglas R. Stinson. *Cryptography: Theory and Practice.* CRC Press LLC, 1995.
36. D. S. Wong, K. Fung, J. K. Liu, and V. K. Wei. On the RS-Code construction of ring signature schemes and a threshold setting of RST. In *5th Intl. Conference on Information and Communication Security (ICICS 2003)*, pages 34–46. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2836.
37. F. Zhang and K. Kim. ID-Based blind signature and ring signature from pairings. In *Proc. ASIACRYPT 2002*, pages 533–547. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2501.