Anonymous Device Authorization for Cellular Networks

Abida Haque * [1], Varun Madathil[2], Bradley Reaves [3], and Alessandra Scafuro[4]

[1]ahaque3@ncsu.edu
[2] vrmadath@ncsu.edu
[3]bgreaves@ncsu.edu
[4]ascafur@ncsu.edu

North Carolina State University

Abstract

Cellular networks connect nearly every human on the planet; they consequently have visibility into location data and voice, SMS, and data contacts and communications. Such near-universal visibility represents a significant threat to the privacy of mobile subscribers. In 5G networks, end-user mobile device manufacturers assign a Permanent Equipment Identifier (PEI) to every new device. Mobile operators legitimately use the PEI to blocklist stolen devices from the network to discourage device theft, but the static PEI also provides a mechanism to uniquely identify and track subscribers. Advertisers and data brokers have also historically abused the PEI for data fusion of location and analytics data, including private data sold by cellular providers.

In this paper, we present a protocol that allows mobile devices to prove that they are not in the blocklist without revealing their PEI to any entity on the network. Thus, we maintain the primary purpose of the PEI while preventing potential privacy violations. We describe a *provably-secure* anonymous proof of blocklist non-membership for cellular network, based on the RSA accumulators and zero-knowledge proofs introduced by Camenisch and Lysyanskaya (Crypto'02) and expanded upon by Li, Li and Xue (ACNS'07). We show experimentally that this approach is viable for cellular networks: a phone can create a blocklist non-membership proof in only 3432 milliseconds of online computation, and the network can verify the proof in less than one second on average. In total this adds fewer than 4.5 seconds to the rare network attach process. This work shows that PEIs can be attested anonymously in 5G and future network generations, and it paves the way for additional advances toward a cellular network with guaranteed privacy.

1 Introduction

Mobile networks provide reliable, high-speed data for applications such as entertainment, IoT deployments, sensitive enterprise communications, and news. A mobile phone is the only device many people have for Internet access, including access to sensitive content [GSM19, Han19]. Despite the need for secure and privacy preserving mobile networks, current cellular networks cannot provide crucial privacy guarantees for location, activity, or identity. Prior work on cellular network privacy issues has primarily focused on external threats to the network over the air interface [DPK+14, DPW16, NSCK17] and investigating malicious external networks [PAS+18].

Recent events have made it clear networks must be designed to provide privacy *even from the subscriber's own network.* Recent disclosures have indicated that providers give or sell location data to law enforcement agents, private individuals, or data brokers [FCC20]. Another concern is the threat of compromised internal infrastructure [VD07], especially by untrusworthy vendors. For example, Western governments have been

*Abida Haque, Varun Madathil and Alessandra Scafuro are supported by NSF grants #1718074,#1764025

concerned by the rising popularity of Huawei as a 5G equipment vendor because of the potential of harm by compromised core network devices.

In 5G, two identifiers are exposed to internal entities: the Subscription Permanent Identifier (SUPI) and the Permanent Equipment Identifier (PEI, also known as the International Mobile Equipment Identity (IMEI) before 5G). Whereas the SUPI is usually stored on a SIM card and identifies a user, the PEI uniquely identifies a User Equipment (UE) device such as a smart phone or cellular modem. The PEI is primarily used to ensure that a phone has not been reported stolen at the time it connects to the network. However, for a data broker who wants to aggregate user profiles, the PEI is ideal to consolidate identities because it is static even across device refreshes and can unify disparate identifying information (e.g., across many email addresses).

In this paper, we present a *provably-secure* protocol for the mobile network operator to confirm that a particular device is not in a global blocklist of lost or stolen devices while not learning the specific device identifier. Our protocol ensures that no proof is linkable to the UE's PEI or to any past or future proof from that UE. Thus, we protect the privacy of user equipment identifiers while still enabling the network to check that only non-blocklisted devices gain access. We demonstrate that our approach is viable by providing a proof-of-concept implementation and reporting performances that are not prohibitive even with modest hardware.

We use cryptographic accumulators and zero-knowledge proofs [LLX07] for private set non-membership. These techniques are well-known, but we are the first to apply them to a cellular context. We formally prove anonymity and soundness in this specific context. We prove that neither eavesdropper nor malicious network can track a user or share their usage data. At the same time, a malicious user cannot craft malicious proofs to gain wrongful access to the network. We then show experimentally that, despite increased computation and data transfer, accumulators with zero-knowledge proofs are sufficiently lightweight to be deployed in mobile networks. Creating and verifying a proof adds in total fewer than 4.5 seconds to the already slow yet infrequent network attach process.

1.1 Motivation

Both the PEI and SUPI are important for a user's identity, but we focus on the PEI for several reasons. First, unlike the SUPI and its predecessors, the PEI (and the IMEI) have historically been abused by third parties for device authentication for advertising and tracking purposes. As a result, the PEI can easily be mapped to a device owner's identity [GZJS12]. Because of the PEI's widespread visibility and linkability, it is in many ways *more* sensitive than the SUPI, which is generally only known by the network. Second, the network requests the PEI less often than the SUPI, and only for a single purpose: testing blocklist membership (elaborated in Sec 8.5). Thus, replacing the PEI with an anonymous credential will be simpler and less performance sensitive than the SUPI. Finally, because the SUPI is required for lawful access requirements (i.e., wiretaps), a privacy-preserving SUPI may not be deployable without legal difficulties. Given these considerations we consider only the private authentication of the PEI. Our work here may also be used as a building block for a privacy-preserving SUPI, which we leave to future work.

We focus on designing our protocols in the 5G setting. We note that our techniques would be equally applicable in earlier generations, though backward compatibility is a challenge to adoption. More importantly, these techniques could be integrated into the next release of 5G or future network standards.

2 Background

2.1 Equipment Identifiers in Cellular Networks

User Equipment (UE): A user's device in 5G is called User Equipment (UE). Each UE is identified by a unique number called the Permanent Equipment Identifier (PEI), previously called the International Mobile Equipment Identity number (IMEI) in prior cellular generations. The PEI is identical to the IMEI in devices that also support LTE, UMTS, or GSM access.

CEIR Maintainer: Each operator maintains a blocklist of PEIs, known as an Equipment Identity Registry (EIR). Each operator contributes their own EIR to a Central Equipment Identity Registry (CEIR) to prevent

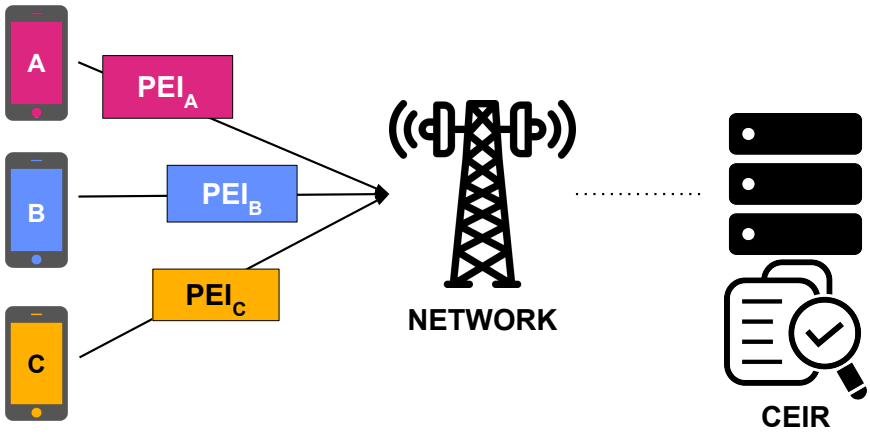

Figure 1: In today's network, a user sends its PEI to the network in the clear. The network then checks if the PEI is in the blocklist or not and authenticates the user.

stolen devices from being resold. The GSMA (a global industry trade organization) currently maintains the CEIR. When an operator authenticates a phone and checks if the phone's PEI is in a blocklist, they compare the PEI to entries in the CEIR.

Network (AMF): When a UE attaches to a cellular network, it transmits messages through a base station (a "gNodeB") to a core network entity, the Access and Mobility Management Function (AMF). Among other tasks, the AMF handles authenticating a UE and checking if a PEI is in a blocklist maintained by the CEIR/EIR.

Manufacturer: The UE manufacturer assigns the PEI at manufacturing time. To ensure global uniqueness, each device model receives a unique PEI prefix, called the Type Allocation Code (TAC). TACs are assigned to manufacturers by the GSMA.

In the current 5G-AKA protocol [3rdb], a UE sends its PEI in the clear to the AMF to prove that it is not blocklisted (Fig. 1). The AMF checks if the PEI is in this list by checking the blocklist maintained by the CEIR. If not, the UE has passed the authentication step.

2.2 Cryptographic Techniques

RSA Accumulator: A cryptographic accumulator represents a set $\mathcal{L}$. The accumulator can be used to check if a particular value is *not* in the set by computing a *non-membership* proof. In our construction we use an RSA accumulator [LLX07].

To compute an RSA accumulator, the owner of a set $\mathcal{L} = \{ \mathsf{id}_1, \ldots, \mathsf{id}_L \}$ multiplies all the elements together as $s = \Pi_{i=1}^{L} \mathsf{id}_i$ and then takes this value to the exponent of a group element $\mathfrak{g}$, so the value is $\mathsf{list\text{-}pk} = \mathfrak{g}^s$. The owner can publish $\mathsf{list\text{-}pk}$ publicly.

Assume there is a party, Alice with identity x who needs credentials to access the network. To receive credentials, her value x is mapped into a prime number id by applying a Hash-to-Prime [GHR99, CMS99, CS00, FT14] function[1] denoted H_{pr}.

Later, to prove that id is not in the set, Alice must prove that id and s are co-prime, i.e., $\gcd(\mathsf{id}, s) = 1$. To achieve this, Alice needs two Bézout coefficients [Béz79] a and b such that $as + b\mathsf{id} = 1$. These values a and b can only exist if s and id are indeed co-prime.

In other words, Alice needs to prove that id is not a factor of one of the exponents of $\mathsf{list\text{-}pk} = \mathfrak{g}^{\mathsf{id}_1 \cdots \mathsf{id}_L} = \mathfrak{g}^s$. Given a and b, anyone can check if the equation $as + b\mathsf{id} = 1$ holds. In the exponent, this check translates to $\mathsf{list\text{-}pk}^a \cdot \mathfrak{g}^{b\mathsf{id}} = \mathfrak{g}$. Note that from the extended GCD algorithm, one can find the GCD and the Bézout coefficients [Béz79].

[1] This is necessary since RSA accumulators work with prime numbers.

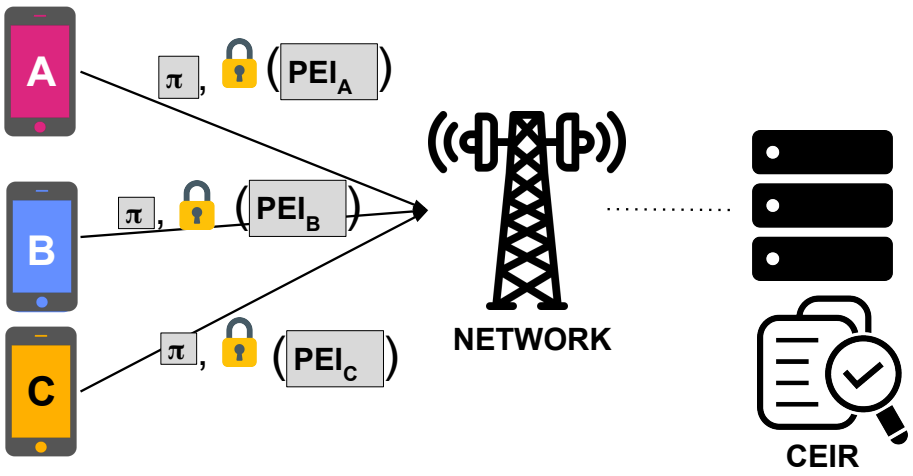

Figure 2: In our scheme, a user sends a proof that it is not in the blocklist and a commitment to its PEI. The network verifies the proof. The PEI is not revealed.

In [LLX07], Alice uses a and B where $B = \mathfrak{g}^{-b}$. To check non-membership, any verifier given (a, B) and list-pk just checks that the equation list-pk^a = $B^{\mathsf{id}}\mathfrak{g}$ is satisfied. Hence, anyone can quickly check non-membership, without having to know the whole set.

Commitment Schemes: We use the Fujisaki-Okamoto (FO) commitment scheme [DF01]. Given two group elements $\mathfrak{g}$ and $\mathfrak{h}$, a committer picks $x \in \mathbb{Z}$ and randomness r. The commitment to x is $c = \mathfrak{g}^x\mathfrak{h}^r$. An FO commitment is secure if given a commitment c it is infeasible to find any solution x and r that opens the commitment, which is true if the Strong-RSA assumption [BP97] holds.

Non-interactive Zero-Knowledge Proof (NIZK): The user needs to prove properties of values (e.g., a value is positive) without revealing the values themselves. At the same time, the user must not be able to create proofs for false statements. A Sigma-protocol is one way to construct such a zero-knowledge (ZK) proof of knowledge.

A Sigma-protocol is a three-move interactive ZK proof scheme where (i) the prover sends a commitment, (ii) the verifier sends a random challenge, and (iii) the prover sends a response. The verifier will compute on the conversation transcript and decide whether to accept or reject. The commitment step "locks in" the prover, so that he can only respond to the challenge correctly if he knows the values.

This interactive protocol can be transformed into a non-interactive with the Fiat-Shamir transform [FS86] where the challenge is computed by the prover as the hash (modeled as a random oracle[PS96]) of the commitment. In one step, the prover now sends the commitments, the computed challenge and the response to the verifer. The verifier also computes the challenge by querying the random oracle and proceeds with the verification as in the interactive protocol.

3 Design Goals

To connect to a cellular network, a mobile device must prove (1) it is operated by an authorized subscriber, and (2) it is not on a blocklist of lost or stolen phones [3rdc, 3rdb]. Rather than sending the PEI to the AMF in the clear, the UE will send a commitment C of the PEI. On each authentication, the UE sends C and a NIZK proof π proving that C is a commitment to a PEI value that is *not* present in the CEIR blocklist (Fig. 2). This high-level approach gives us the security properties of anonymity and soundness (explained in Sect. 6.1), but we need to ensure that our solution is practical. We require the following:

The proof size must be small: The statement that a UE is proving in zero-knowledge is "PEI is not in CEIR's blocklist". In a naïve approach the size of the ZK proof is proportional to the size of the entire blocklist, which can have millions of devices. Therefore, we replace the public CEIR blocklist in the current network

with a public accumulator list-pk with the set holder as the CEIR maintainer. Accumulators are explained in Section 2.2.

The proof must be fast: In our scheme, a UE has a credential on its PEI that it uses to prove non-membership in an RSA accumulator. We use RSA accumulators [BDM93] and Sigma-protocols for RSA accumulators ([LLX07]) so that both proof size and proving time are *constant* with respect to the size of the set. Furthermore, the accumulator can be updated with one operation per each new element[2]. While there are many ways to create non-membership proofs [Mer88, Ngu05, DT08], we choose RSA accumulators because the size of the blocklist grows regularly, and we need to ensure that updates are efficient.

Frequent blocklist updates must not impact performance: The blocklist is updated regularly through the combined effort of mobile network providers that share their EIRs. In our setting, the CEIR maintainer will need to update the accumulator as well. Each update to the accumulator requires the UE to update its credential.

The UE needs to sequentially update its credentials for each addition to the blocklist because batch updates for accumulators are impossible [CH10, BCD+17]. This may not be practical for a fast-growing blocklist. To mitigate this issue, we partition the CEIR based on the leading digits of the PEI, which are the TAC indicating the manufacturer and device model. The TAC is eight decimals, so only a fraction of the TAC will be used for partitioning the blocklist. Then when a UE attempts to prove non-membership, it must also announce on which blocklist she has a non-membership credential. This will reduce the quantity of updates done daily by a phone but limits the anonymity set from, e.g., "all phones" to "all Samsung phones". We present the protocol without describing any partitions.

We expect that credential updates will be distributed over the Internet from a common source used by all devices. Also, a UE can perform credential update calculations in the background, rather than at verification time.

The protocol must limit replay attacks: An adversary that eavesdrops on the connection between the network and the phone could "steal" a valid proof and then use it to create a proof that looks as though it came from her own (blocklisted) phone.

To limit the adversary's success of mounting the attack, we use two techniques. First, we embed the numeric identifier for the serving cellular network and the cell ID of the gNodeB to prevent reuse of a proof in another location in the network. Second, we introduce a *window of validity* for each proof. Each UE embeds a timestamp in the proof that marks the time of proof creation. Upon receipt of an authentication request, the network checks if the proof is in the window of validity depending on the time stamp. If not, the network rejects the proof. In this way, we limit the power of the adversary by reducing the time over which it can perform a replay attack. As the time stamp is embedded in the proof, a malicious UE cannot change the time stamp without invalidating the proof.

No cryptographic solution can entirely prevent the cloning of legitimate device credentials to a stolen device, but tamper-proof hardware could reduce the risk. Indeed, IMEI cloning is already possible in some devices, and we do not claim that our protocol will be secure against credential cloning. A particularly strong adversary could duplicate the credentials for a proof from a non-blocklisted phone to a modified block-listed one; depending on the security architecture of the device, this could be very challenging. For example, Qualcomm modems have a secure boot process requiring signed firmware. We do prove in the following sections that an attacker cannot fabricate new credentials.

4 Zero-Knowledge Protocol

In this section, we describe each party involved in the protocol, their role, and the procedure they execute.

CEIR maintainer. The CEIR maintainer maintains the blocklist and publishes the corresponding accumulator. For simplicity, in our scheme, this entity also assigns PEI to each UE. The CEIR maintainer executes procedures Setup, Enroll, UpdBList, that we informally describe below. A formal description of these procedures is provided in Fig. 4.

[2]We avoid using general accumulators (e.g., based on hash functions) and general purpose ZK proofs (e.g., SNARKs).

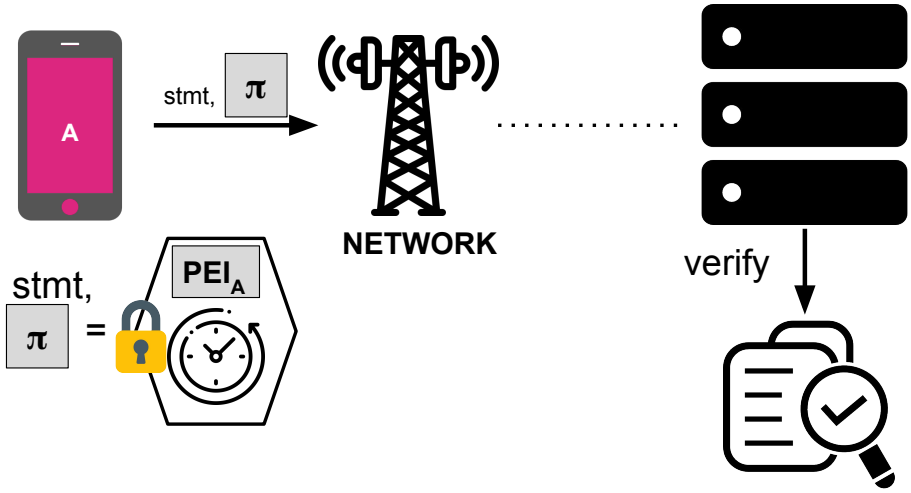

Figure 3: Authentication and Verification. A user uses their pei and a timestamp tms to create a proof that they send to the network. The network can then verify if the user has a valid pei.

Setup. The CEIR maintainer sets up the system parameters that are used by the other entities. It samples and publishes the secure parameters for the RSA and commitment schemes. Using the RSA parameters, the CEIR maintainer initializes the accumulator value (representing an empty blocklist) using a secret seed. The secret seed ensures that the CEIR maintainer is the only entity who can give credentials to users in the Enroll procedure.

Enroll. This algorithm assigns each UE an ID and a credential it can use to prove that it is not in a blocklist. Specifically, for a UE that comes with PEI pei, the CEIR maintainer first maps pei onto a prime id. Next, it will create non-membership credentials for the id using the current RSA accumulator. It will return the id and credential to the UE.

Update blocklist. This algorithm updates the blocklist when a UE's PEI has been revoked. To blocklist, the CEIR maintainer adds an id to the accumulator and creates a signature on the updated revoked identity id^* to send to all the users.

User (UE). The UE receives an id and a credential from the CEIR maintainer at initialization. The UE need not interact with CEIR maintainer afterwards. The UE runs Auth (which is randomized) to prove that it is not in the blocklist, UpdCred to update the credential when the blocklist is updated, and VrfyUpdBList that verifies the authenticity of the blocklist's update. These are described in Fig. 5

Authenticate The UE authenticates its pei by calculating a ZK proof of non-membership using the id, the non-membership credential, and the most updated accumulator. This proof shows that the UE has knowledge of a credential that belongs to an id that has not been accumulated (i.e., not in the blocklist). The UE sends the non-membership proof to the AMF. We illustrate the flow in Fig 3.

Update Credential This algorithm updates a UE's credential upon receipt of a new blocklist. The CEIR maintainer updates the blocklist (here, the accumulator list-pk) with the PEI when a UE is reported stolen. Only the CEIR maintainer may update the blocklist which is enforced by a signature on the upd. The UE verifies this signature and updates its credentials using the updated accumulator and its old credentials.

Network (AMF) Upon receipt of a user's proof, the AMF executes the procedure Vrfy (Fig. 7) to see if the UE is blocklisted using the most updated accumulated blocklist.

Verify. This algorithm authenticates a proof that the PEI is not in the blocklist. Upon receipt of the set of commitments and a proof from a UE, the AMF will parse the commitments and proof. The AMF then verifies the proof on the current accumulator $\mathsf{list\text{-}pk}^*$ and with the time stamp tms. Specifically, the AMF checks that the proof was received at a time within $\mathsf{tms} + \mathsf{Expiry}$. If the commitments are consistent with one another, and the proof verifies, the AMF will accept. Else, it will reject. We note that even if the AMF is completely malicious and colludes with the CEIR maintainer, it cannot learn the PEI of the UE that is trying to authenticate.

$\underline{\mathsf{Setup}(1^\lambda)}$

1. Select $\lambda-$bit safe primes $p = 2p' + 1, q = 2q' + 1$; let $n = pq$
2. Sample $\mathsf{a} \leftarrow_\$ \mathbb{Z}_n^*$
3. Set $\mathfrak{g} = \mathsf{a}^2 \mod n$.
4. Set $t = 128, m = 80$
5. Sample $\alpha \leftarrow_\$ \mathbb{Z}_n^*$
6. Set $\mathfrak{h} = \mathfrak{g}^\alpha$
7. Define $\mathsf{pp} = (\mathfrak{g}, \mathfrak{h}, n, t, m)$
8. Sample $r' \leftarrow\$ \{0,1\}^\eta$.
9. $r = \mathsf{H}_{\text{pr}}(r')$.
10. Define $\mathsf{list\text{-}sk} := (p', q', r)$
11. Define $\mathsf{list\text{-}pk} = \mathfrak{g}^r$
12. Initialize $\mathcal{L} = \emptyset$
13. Compute $\mathsf{mpk}, \mathsf{msk} = \mathsf{SS.Gen}(1^\lambda)$
14. Output $(\mathcal{L}, \mathsf{msk}, \mathsf{mpk}, \mathsf{list\text{-}pk}, \mathsf{list\text{-}sk}, \mathsf{pp})$

$\underline{\mathsf{UpdBList}(\mathsf{pp}, \mathsf{list\text{-}pk}, \mathcal{L}, \mathsf{id}, \mathsf{msk})}$

1. If $\mathsf{id} \in \mathcal{L}$: return "Already Revoked"
2. Else:
   (a) $\mathcal{L} := \mathcal{L} \cup \{\mathsf{id}\}$
   (b) Update $\mathsf{list\text{-}pk} = \mathsf{list\text{-}pk}^{\mathsf{id}}$
   (c) $\mathsf{sigUpmsg} \leftarrow \mathsf{SS.Sign}(\mathsf{msk}, \mathsf{list\text{-}pk}, \mathsf{id})$
   (d) Set $\mathsf{upd} := ((\mathsf{list\text{-}pk}, \mathsf{id}), \mathsf{sigUpmsg})$
   (e) return upd

$\underline{\mathsf{Enroll}(\mathsf{pp}, \mathsf{list\text{-}sk}, \mathcal{L}, \mathsf{pei})}$

1. Parse $\mathsf{list\text{-}sk} = (p', q', r)$.
2. Sample random $\mathsf{nonce} \leftarrow \mathbb{Z}_n$
3. $\mathsf{id} := \mathsf{H}_{\text{pr}}(\mathsf{pei} \| \mathsf{nonce})$.
4. If $\mathsf{id} = 1 \mod \phi(n)$, then run Step 3 with $\mathsf{nonce} = \mathsf{nonce} + 1$, else continue.
5. $s^* \leftarrow r \cdot \prod_{s \in \mathcal{L}} s$.
6. $a, b \leftarrow \mathsf{Bézout}(s^*, \mathsf{id})$.
7. $B \leftarrow g^{-b}$
8. return $\mathsf{id}, \mathsf{sk}_{\mathsf{id}} = (a, B)$

Figure 4: CEIR Maintainer's Functions.

$\underline{\mathsf{Auth}(\mathsf{pp}, \mathsf{list\text{-}pk_{id}}, \mathsf{tms}, \mathsf{id}, \mathsf{sk_{id}})}$

1. Parse $\mathsf{sk_{id}} = (a, B)$.
2. Set $\mathsf{stmt} = (\mathsf{list\text{-}pk_{id}}, \mathsf{tms})$.
3. $\mathsf{Wit} = \{\mathsf{id}, a, B\}$
4. Output $(\mathsf{stmt}, \mathsf{proof})$ = $\mathsf{ProveS}(\mathsf{pp}, \mathsf{stmt}, \mathsf{Wit})$ (See Fig. 6)

$\underline{\mathsf{VrfyUpdBList}(\mathsf{mpk}, \mathsf{list\text{-}pk}, \mathsf{id}, \mathsf{sigUpmsg})}$

1. If $1 \leftarrow \mathsf{SS.Vrfy}(\mathsf{mpk}, \mathsf{list\text{-}pk}, \mathsf{id}, \mathsf{sigUpmsg})$ return 1.

$\underline{\mathsf{UpdCred}(\mathsf{pp}, \mathsf{mpk}, \mathsf{list\text{-}pk_{id}}, \mathsf{upd}, \mathsf{sk_{id}}, \mathsf{id})}$

1. Parse $\mathsf{upd} = ((\mathsf{list\text{-}pk}^*, \mathsf{id}'), \mathsf{sigUpmsg})$
2. If $1 \leftarrow \mathsf{VrfyUpdBList}(\mathsf{mpk}, \mathsf{list\text{-}pk}^*, \mathsf{id}', \mathsf{sigUpmsg})$ return $\perp$. Else continue.
3. Compute $(a_0, r_0) = \mathsf{Béz}(\mathsf{id}', \mathsf{id})$
4. Set $a' = aa_0 \mod \mathsf{id}$
5. Compute $r = (a' \cdot \mathsf{id}' - a) \cdot \mathsf{id}^{-1} \mod n$
6. Set $B' = \mathsf{list\text{-}pk}_{\mathsf{id}}^r \cdot B \mod n$
7. Return $\mathsf{list\text{-}pk}^*$, $\mathsf{sk}'_{\mathsf{id}} = (a', B')$

Figure 5: UE Functions

5 Session Resumption

In this section, we describe a session resumption protocol that reduces authorization latency by permitting a UE to show proof of non-membership based on a previous execution of our zero-knowledge protocol. At a high level, this will involve the provider issuing a token to the UE that can later be checked to confirm authorization. In such a protocol, there are several tradeoffs. To improve performance, both the UE and the provider will want the token to have a long period of validity to reduce the number of ZK proofs required. For security, the provider will want to have a shorter period of validity to prevent a blocklisted device from continuing to attach to the network. While the provider will not be able to learn the PEI, the UE will want a shorter period lifetime to reduce linkability of behavior to a single identifier.

We now describe how the session resumption T is computed. At the end of 5G-AKA protocol, the network and the UE have agreed on a security context that includes keys and protocols for confidentiality (k_c) and integrity, and the network has verified the ZK proofs and authorized that this UE has a valid PEI. The network computes the token $T = H(\mathsf{proof}, \mathsf{salt})$, where H is a hash function (e.g., SHA-256) and salt is a random string. The network also sets an expiration time $\mathsf{RExpiry}$. The network will store $(T, \mathsf{salt}, \mathsf{proof}, \mathsf{RExpiry})$ along with other attributes of the UE.

Once a token is created, the network then sends $\mathsf{Enc}(T, \mathsf{salt}, \mathsf{RExpiry})$ to the UE encrypted under the k_c from the AKA protocol. The UE decrypts the message and verifies that $T = H(\mathsf{proof}, \mathsf{salt})$. The UE must be able to verify the creation of the token because if the token were opaque, a malicious network could embed linking information in it. The UE can use the token T upon receiving a PEI request from the network. Each time the network receives a token T, the network checks if the current time is earlier than $\mathsf{RExpiry}$ for the current token, and if so, it accepts the token and authorizes the UE.

Tokens are linkable but not truly anonymous. The network can see that the same token was used for different authorizations. However, the token reveals no further information about the PEI, so an adversary can still not de-anonymize the user. Also, no malicious UE can create or copy an honest UE's token because the token is created as $T = H(\mathsf{proof}, \mathsf{salt})$. Even if an honest proof is known to a malicious UE, it cannot learn the salt since it is encrypted using k_c. The token is also encrypted using the same key. Thus, the malicious UE will need to break session encryption to learn the token T, which is currently not known to be feasible.

We note that $\mathsf{RExpiry}$ is set by the network. A network could set a longer expiration than the user would be comfortable with, or a malicious network could attempt to use expiration time as a channel to link PEI

$\underline{\mathsf{ProveS}(\mathsf{pp}, \mathsf{stmt}, \mathsf{Wit})}$:

Parse $\mathsf{pp} = \mathfrak{g}, \mathfrak{h}, n, t, m, \quad \ell = |n|/2$

Parse $(\mathsf{list\text{-}pk}_{\mathsf{id}}, \mathsf{tms}) = \mathsf{stmt}, (\mathsf{id}, a, B) = \mathsf{Wit}$

Sample $w, r_{\mathsf{id}}, r_a, r_w, r_z, r_e \leftarrow \{0,1\}^{|n|}$

$c_{\mathsf{id}} = \mathfrak{g}^{\mathsf{id}}\mathfrak{h}^{r_{\mathsf{id}}} \mod n, c_a = \mathfrak{g}^{a}\mathfrak{h}^{r_a} \mod n$

$c_B = B\mathfrak{g}^{w} \mod n, c_e = (c_B)^{\mathsf{id}}\mathfrak{h}^{r_e} \mod n$

$c_w = \mathfrak{g}^{w}\mathfrak{h}^{r_w} \mod n$

$z = \mathsf{id} \cdot w \mod n, c_z = \mathfrak{g}^{z}\mathfrak{h}^{r_z} \mod n$

$\mathsf{PoK}_1 \leftarrow \mathsf{EqCom}(\mathsf{X} = (c_e, c_{\mathsf{id}}, c_B, \mathfrak{h}, \mathfrak{g}, \mathfrak{h}, n, n, \mathsf{stmt}), \mathsf{W} = (\mathsf{id}, r_e, r_{\mathsf{id}}))$

$\mathsf{PoK}_2 \leftarrow \mathsf{AccRel}(\mathsf{X} = (c_e, c_a, c_z, c_B, \mathfrak{g}, \mathfrak{h}, n, \mathsf{stmt}), \mathsf{W} = (\mathsf{id}, r_e, a, r_a, z, r_z))$

$\mathsf{PoK}_3 \leftarrow \mathsf{ProdCom}(\mathsf{X} = (c_z, c_w, c_{\mathsf{id}}, \mathfrak{g}, \mathfrak{h}, n, \mathsf{stmt}), \mathsf{W} = (r_z, w, r_w, \mathsf{id}, r_{\mathsf{id}}))$

$\mathsf{PoK}_4 \leftarrow \mathsf{RangeProof}(\mathsf{X} = (c_{\mathsf{id}}, t, m, \mathfrak{g}, \mathfrak{h}, n, \mathsf{stmt}), \mathsf{W} = (\mathsf{id}, r_{\mathsf{id}}))$

$\mathsf{PoK}_5 \leftarrow \mathsf{RangeProof}(\mathsf{X} = (c_a, t, m, \mathfrak{g}, \mathfrak{h}, n, \mathsf{stmt}), \mathsf{W} = (a + 2^{\ell}, r_a))$

Return $c_{\mathsf{id}}, c_a, c_B, c_w, c_z, c_e, \mathsf{PoK}_2 \ldots \mathsf{PoK}_5$

Figure 6: Proving Algorithm (composition of Sigma-protocols described in Appendix A)

$\underline{\mathsf{Vrfy}(\mathsf{pp}, \mathsf{list\text{-}pk}^*, \mathsf{stmt}, \mathsf{proof}, \mathsf{Expiry})}$

1. Parse $\mathsf{stmt} = (\mathsf{list\text{-}pk}_{\mathsf{id}}, \mathsf{tms})$
2. if $\mathsf{time} - \mathsf{tms} > \mathsf{Expiry}$ then reject.
3. Check $\mathsf{list\text{-}pk}^* = \mathsf{list\text{-}pk}_{\mathsf{id}}$
4. Return $\mathsf{zk.Ver}(\mathsf{pp}, \mathsf{stmt}, \mathsf{proof})$

Figure 7: Network functions

authorizations. We require two simple mechanisms to prevent these attacks. First, we mandate that the expiration time must be a deterministic function of the token. Second, we mandate that the protocol allow the UE to re-attach using a full ZK proof at anytime, and that the UE should choose to re-attach at a random time before expiration.

The protocol is stateful, but storage is feasible. 350 million subscribers (more than the population of the United States) each with a 32-byte (length of SHA256), would require merely 11.2GB to be stored. Tokens can be stored in the Structured Data Storage Function, a database service included in the 5G architecture.

6 Framework and Security Definitions

In this section, we define the syntax, oracles, and finally, the security definitions and games. We will first define the syntax of our protocol for blocklist non-membership proofs. In Section 7 we prove that our protocol meets the security goals.

Definition 1 (Anonymous blocklist non-membership protocol)**.** An anonymous blocklist non-membership

$\underline{\mathsf{zk.Ver}(\mathsf{pp}, \mathsf{stmt}, \mathsf{proof})}$

Parse $\mathsf{stmt} = (\mathsf{list\text{-}pk}_{\mathsf{id}}, \mathsf{tms})$

Parse $c_{\mathsf{id}}, c_a, c_B, c_w, c_z, c_e, \mathsf{PoK}_2 \ldots \mathsf{PoK}_5 \leftarrow \mathsf{proof}$

Parse $(\mathfrak{g}, \mathfrak{h}, n, t, m) \leftarrow \mathsf{pp}$

Compute $k = |n|$

$\mathsf{VrfyEqCom}(\mathsf{X} = (c_e, c_{\mathsf{id}}, c_B, h, \mathfrak{g}, \mathfrak{h}, n, n, \mathsf{stmt}), \mathsf{PoK}_1)$

$\mathsf{VrfyAccRel}(\mathsf{X} = (c_e, c_a, c_z, c_B, \mathfrak{g}, \mathfrak{h}, n, \mathsf{stmt}), \mathsf{PoK}_2)$

$\mathsf{VrfyProdCom}(\mathsf{X} = (c_z, c_w, c_{\mathsf{id}}, \mathfrak{g}, \mathfrak{h}, n, k), \mathsf{PoK}_3)$

$\mathsf{VrfyRangeProof}(\mathsf{X} = (c_{\mathsf{id}}, t, m, \mathfrak{g}, \mathfrak{h}, n, \mathsf{stmt}), \mathsf{PoK}_4)$

$\mathsf{VrfyRangeProof}(\mathsf{X} = (c_a, t, m, \mathfrak{g}, \mathfrak{h}, n, \mathsf{stmt}), \mathsf{PoK}_5)$

Figure 8: Verification algorithms of Sigma-protocols in Appendix A

protocol BListNM is a tuple $\Pi = (\mathsf{Setup}, \mathsf{Enroll}, \mathsf{UpdBList}, \mathsf{VrfyUpdBList}, \mathsf{UpdCred}, \mathsf{Auth}, \mathsf{Vrfy})$ executed between three parties: a CEIR maintainer, a user (UE) and a verifier (AMF).

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathcal{L}, \mathsf{msk}, \mathsf{mpk}, \mathsf{list\text{-}pk}, \mathsf{list\text{-}sk}, \mathsf{pp})$ On input security parameter 1^λ produces a tuple where $\mathcal{L} = \emptyset$, msk, mpk are key pairs of the CEIR maintainer, list-pk is the public key associated to the blocklist, list-sk is the corresponding secret key, and pp are the public parameters.
- $\mathsf{Enroll}(\mathsf{pp}, \mathsf{list\text{-}sk}, \mathcal{L}, x) \rightarrow (\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ On input x (the PEI), the CEIR maintainer produces an identity id and an associated secret credential $\mathsf{sk}_{\mathsf{id}}$.
- $\mathsf{UpdBList}(\mathsf{pp}, \mathsf{list\text{-}pk}, \mathcal{L}, \mathsf{id}, \mathsf{msk}) \rightarrow (\mathcal{L}^*, \mathsf{list\text{-}pk}^*, \mathsf{upd})$ On input id, the CEIR maintainer updates the list-pk to $\mathsf{list\text{-}pk}^*$ and creates an upd using its secret key msk and sends to all the users.
- $\mathsf{VrfyUpdBList}(\mathsf{pp}, \mathcal{L}^*, \mathsf{list\text{-}pk}^*, \mathsf{upd}, \mathsf{mpk}) \rightarrow 0/1$ On receiving upd from the CEIR maintainer, a user verifies that the signature of upd that was created by the CEIR maintainer using mpk.
- $\mathsf{UpdCred}(\mathsf{pp}, \mathsf{list\text{-}pk}^{\mathsf{id}}, \mathsf{list\text{-}pk}^*, \mathsf{upd}^*, \mathsf{sk}_{\mathsf{id}}, \mathsf{id}) \rightarrow (\mathsf{sk}^*_{\mathsf{id}}, \mathsf{list\text{-}pk}^{\mathsf{id}})$ Upon receiving an update message upd, and the updated public key list-pk, the user updates its credentials to get a new $\mathsf{sk}^*_{\mathsf{id}}$. Users who have not been revoked need to update their credentials with respect to the $\mathsf{list\text{-}pk}^*$.
- $\mathsf{Auth}(\mathsf{pp}, \mathsf{stmt}, \mathsf{cred}) \rightarrow \mathsf{proof}$: The user parses $\mathsf{stmt} = (\mathsf{list\text{-}pk}_{\mathsf{id}}, \mathsf{tms})$ and $\mathsf{cred} = (\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$. The user uses his secret credentials $\mathsf{sk}_{\mathsf{id}}$ and id to create a proof that he is not in the blocklist represented by $\mathsf{list\text{-}pk}_{\mathsf{id}}$. Finally, tms is a timestamp that is used to enforce non-replayability of the proof.
- $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{list\text{-}pk}, \mathsf{stmt}, \mathsf{proof}) \rightarrow 0/1$: On input statement stmt and proof proof outputs $b = 1$ if it is convinced that the user who computed the proof is not in the blocklist list-pk and $b = 0$ otherwise.

6.1 Trust Assumptions

Adversary: We assume that the adversary can see all network messages related to the PEI authentication and can use this information to try to link authentication requests to UEs. This strong adversary can corrupt the AMFs and see the non-membership proofs provided by any UE in the system. We guarantee that even if an adversarial network tries to target a UE, it cannot de-anonymize the UE.

The adversary can also corrupt the CEIR maintainer and learn all the PEIs and credentials that the CEIR maintainer has given to the UEs. This strong adversary models the realistic scenario where the CEIR maintainer is coerced to reveal information about a specific UE. We want to guarantee that anonymity is preserved even in

such a case. However, we do require that the CEIR runs both the Setup algorithm and the UpdBList algorithm honestly.

In our protocol, when a CEIR maintainer adds an ID to the blocklist, it broadcasts them. While this means the adversary can collect a list of the blocklisted IDs (as opposed to in the current protocol where these IDs are *not* broadcast, this does not affect the security of learning *which* PEI is authenticating.

Security Goals: In presence of this adversary, the protocol described Section 4 achieves the following security goals: anonymity (Def. 2), soundness (Def. 3). non-replayability, and revocation unforgeability (Def. 4).

Formal Definition of the Adversarial Power: The adversary's power is formally captured by *oracles* that the adversary can query during her attack. We assume that the adversary has only these capabilities and does not mount other attacks like blocking the network. The adversary is probabilistic polynomial time (PPT), which means it runs in polynomial time and may use randomness.

The set of oracles is $\mathcal{O} = (\mathsf{Enroll}, \mathsf{UpdBList}, \mathsf{Corrupt}, \mathsf{Auth})$. We use the notation $\mathcal{A}^{\mathcal{O}}$ to say that the adversary has access to all the oracles. There are global sets maintained by the oracles: $\mathcal{H}$: the set of honest users, $\mathcal{L}$: set of blocklisted users, $\mathcal{M}$: set of users corrupt by the adversary. These are initialized to $\emptyset$.

1. $\mathsf{Enroll}(x) \rightarrow \mathsf{id}$. This oracle allows the adversary to add an honest user to the system. On input x, the adversary can learn an id. The oracle updates the set of honest users as $\mathcal{H} := \mathcal{H} \cup \{\mathsf{id}\}$.

2. $\mathsf{UpdBList}(\mathsf{id}) \rightarrow \mathcal{L}^*, \mathsf{list\text{-}pk}^*, \mathsf{upd}$. This oracle allows the adversary to revoke a certain id by adding it to the blocklist. The input to the oracle is id and the output of this oracle is the updated $\mathcal{L}^*$, $\mathsf{list\text{-}pk}^*$ and upd. This oracle updates the set of revoked identities $\mathcal{L} := \mathcal{L} \cup \{\mathsf{id}\}$.

3. $\mathsf{Corrupt}(\mathsf{id}) \rightarrow \mathsf{sk}_{\mathsf{id}}$. The oracle allows the adversary to learn a user's secret key (if id has not been revoked). The oracle uses Enroll to calculate the $\mathsf{sk}_{\mathsf{id}}$ of the user and returns it to the adversary. The set of corrupted users $\mathcal{M}$ is updated as $\mathcal{M} := \mathcal{M} \cup \{\mathsf{id}, \mathsf{sk}_{\mathsf{id}}\}$. The set of honest users is also updated by removing the identity as $\mathcal{H} := \mathcal{H} \setminus \{\mathsf{id}\}$.

4. $\mathsf{Auth}(\mathsf{list\text{-}pk}, \mathsf{id}) \rightarrow \pi$. This oracle allows the adversary to get a proof on an id. If $\mathsf{id} \notin \mathcal{L}$, $\mathsf{id} \in \mathcal{H} \cup \mathcal{M}$, then the oracle returns a proof π, otherwise returns $\perp$. The requested proofs are stored in a set $\mathcal{P}$. On each request, the set is updated as $\mathcal{P} := \mathcal{P} \cup \{(\mathsf{id}, \pi)\}$.

Anonymity. Anonymity means that the network (AMFs) and CEIR maintainer cannot learn any information about the PEI of a UE that provides an accepting proof. The adversary here can observe many proofs from the UEs. Finally, they pick two ids that are not blocklisted (i.e., not in set $\mathcal{L}$) and receive a proof on one of those ids at random. The adversary wins anonymity if it can guess which id it received a proof for. Even if a malicious network or CEIR observes all the proofs computed by the UEs and knows the PEI and secret credentials of the UEs, it cannot link a proof to a specific PEI.

In the security game, where after observing many proofs from the UEs, the adversary picks ids of two honest users and sends them to the challenger. The challenger selects one of the ids at random and sends the proof corresponding to that id. The adversary $\mathcal{A}$ wins the game if it guesses correctly. Even if the adversary corrupted all but two UEs, she cannot tell which of the two honest UEs the proof corresponds to with a probability better than $\frac{1}{2}$. Fig 9 illustrates this definition.

Anonymity implies unlinkability of proofs. An adversary playing the anonymity game creates proofs for id_0 by querying the Auth oracle before sending id_0 and id_1. Upon receiving π_b, if the adversary can link π_b with a proof of id_0 then it outputs 0, and otherwise it outputs 1. This way, the adversary wins the anonymity game if it can link two messages.

Definition 2. (Anonymity) A BListNM protocol $\Pi = (\mathsf{Setup}, \mathsf{Enroll}, \mathsf{UpdBList}, \mathsf{Vrfy}, \mathsf{UpdBList}, \mathsf{UpdCred}, \mathsf{Auth}, \mathsf{Vrfy})$ satisfies anonymity if, for any PPT adversary $\mathcal{A}$ it holds:

$$Pr[\mathsf{GameAnon}^{\Pi}_{\mathcal{A}}(\lambda) = 1] < \mathsf{negl}(n).$$

Soundness. Soundness means that a user without valid credentials cannot prove that it is not in the blocklist. More specifically, a user cannot authenticate using blocklisted PEIs, create non-membership proofs, or generate

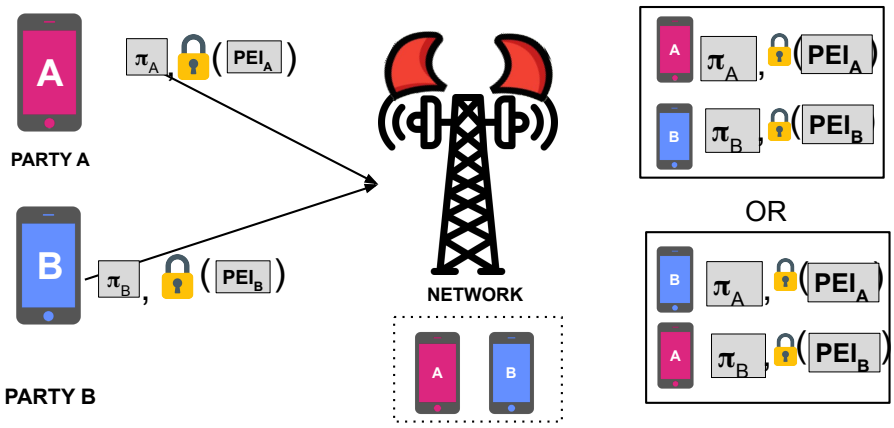

Figure 9: A malicious network upon receiving a proof cannot distinguish if the proof corresponds to pei_A or pei_B even when it is given the credentials and PEIs of all UEs.

Game 1: $\mathsf{GameAnon}_{\mathcal{A}}^{\Pi}(\lambda)$

$(\mathcal{L}, \mathsf{msk}, \mathsf{mpk}, \mathsf{list\text{-}pk}, \mathsf{list\text{-}sk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^{\lambda})$

$\mathsf{id}_0, \mathsf{id}_1 \leftarrow \mathcal{A}^{\mathcal{O}}(\mathcal{L}, \mathsf{list\text{-}pk}, \mathsf{mpk}, \mathsf{pp})$

Compute latest $\mathsf{list\text{-}pk}^*$ according to $\mathcal{L}$

if $\mathsf{id}_0, \mathsf{id}_1 \notin \mathcal{L}$

$b \leftarrow \{0, 1\}$

$\pi_b \leftarrow \mathsf{Auth}(\mathsf{pp}, (\mathsf{list\text{-}pk}^*, \mathsf{tms}), (\mathsf{id}_b, \mathsf{sk}_{\mathsf{id}_b}))$

$b' \leftarrow \mathcal{A}(\pi_b)$

if $b = b'$, return 1

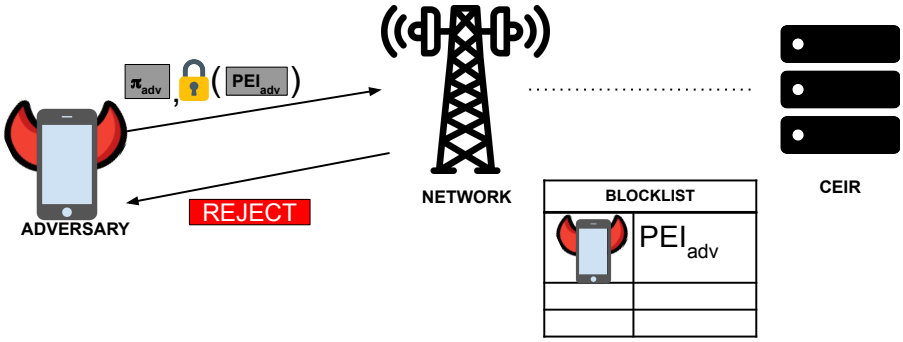

Figure 10: A malicious blocklisted UE cannot provide a valid proof to convince the network that it is not in the blocklist

valid credentials without knowing a PEI, even if this user has observed and collected proofs from other UEs. Fig 10 shows any UE whose PEI is in the blocklist will be rejected.

Definition 3. (Soundness) A BListNM protocol $\Pi = (\mathsf{Setup}, \mathsf{Enroll}, \mathsf{UpdBList}, \mathsf{Vrfy}, \mathsf{UpdBList}, \mathsf{UpdCred}, \mathsf{Auth}, \mathsf{Vrfy})$ satisfies soundness if there exists a PPT extractor Ext such that for any PPT adversary $\mathcal{A}$ participating in Game 2, there exists a negligible function negl such that: $Pr[\mathsf{GameSound}_{\mathcal{A}}^{\Pi}(\lambda) \rightarrow 1] \leq \mathsf{negl}(\lambda)$

Game 2: $\mathsf{GameSound}^{\Pi}_{\mathcal{A}}(\lambda)$

$(\mathcal{L}, \mathsf{msk}, \mathsf{mpk}, \mathsf{list\text{-}pk}, \mathsf{list\text{-}sk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$
$\pi^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{list\text{-}pk}, \mathsf{mpk}, \mathsf{pp}, \mathcal{L})$
If $\pi^* \in \mathcal{P}$ return 0
If $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{list\text{-}pk}^*, \pi^*) = 0$, return 0
$(\mathsf{id}^*, \mathsf{sk}^*_{\mathsf{id}}) \leftarrow \mathsf{Ext}(\pi^*)$
If $\mathsf{id}^* \in \mathcal{L}$ or $\mathsf{id}^* \in \mathcal{H}$ return 1.

Non-replayability. Non-replayability requires that a malicious UE cannot pass a non-membership proof by replaying a proof computed by an honest UE. The non-replayability game mirrors the soundness game but considers additional application-based tags that make the proof unique. To win the game, the adversary must provide a (potentially old) proof with respect to a fresh tag.

Revocation Unforgeability. Revocation unforgeability guarantees that only the CEIR maintainer can add users to the blocklist.

The adversary here does not know the CEIR maintainer's private information but can add users, corrupt them, and see proofs from the users. The adversary then tries to create an update message that will be accepted by a user.

Definition 4. (Revocation Unforgeability) A BListNM protocol $\Pi = (\mathsf{Setup}, \mathsf{Enroll}, \mathsf{UpdBList}, \mathsf{VrfyUpdBList}, \mathsf{UpdCred}, \mathsf{Auth}, \mathsf{Vrfy})$ satisfies revocation unforgeability if for any PPT adversary $\mathcal{A}$s:

$$Pr[\mathsf{GameRevForge}^{\Pi}_{\mathcal{A}}(\lambda) = 1] < \mathsf{negl}(n)$$

Game 3: $\mathsf{GameRevForge}^{\Pi}_{\mathcal{A}}(\lambda)$

$(\mathcal{L}, \mathsf{msk}, \mathsf{mpk}, \mathsf{list\text{-}pk}, \mathsf{list\text{-}sk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathcal{L}^*, \mathsf{list\text{-}pk}^*, \mathsf{upd}^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathcal{L}^*, \mathsf{list\text{-}pk}^*)$
$(\text{if } \cdot, \cdot, \mathcal{L}^*, \mathsf{list\text{-}pk}^*, \mathsf{upd}^*) \notin \mathcal{L} \wedge$
$\quad 1 \leftarrow \mathsf{VrfyUpdBList}(\mathsf{pp}, \mathcal{L}^*, \mathsf{list\text{-}pk}^*, \mathsf{upd}^*, \mathsf{mpk})$
$\quad$ return 1

7 Proofs of Security

Theorem 1. *If the Fujisaki-Okamoto commitment scheme is hiding, H is a random oracle, Sigma-protocols* EqCom, AccRel, ProdCom, *and* RangeProof *(Sect. A) satisfy zero-knowledge ([BS20] Def. 19.5), then protocol* Π *satisfies anonymity (Def. 2).*

Intuition. Recall that in Game 1 the adversary $\mathcal{A}$ picks two identities $\mathsf{id}_0, \mathsf{id}_1$ and receives a proof π_b on one of the id's and $\mathcal{A}$ wins by guessing b. We prove that when Game 1 is instantiated with protocol BListNM (Fig. 6), the two cases π_0 and π_1 are indistinguishable.

We use a sequence of hybrids to switch from having the adversary receive a proof for π_0 to π_1 and show each change is indistinguishable. ZK means that an honestly generated proof cannot be distinguished from a simulated proof. Thus, we start by switching from honest proofs to simulated proofs. Next, by using the hiding properties of commitments, we switch the commitments from $b = 0$ to $b = 1$. Finally, we switch back to using honest proofs for $b = 1$. The formal proof with all hybrid arguments is provided in Appendix A.2. □

Table 1: Time for Network to Verify, Revoke, and Enroll (ms)

Name	Mean	Std. Dev.
Enroll[3]	939	2.65
Verify Time	982	85.6
Revoked Time	13	0.33

Table 2: UE Battery Use

Name	Std. Dev.	Charge (mA·sec)
100 Auth	3.2	544.95
5 Update	3.2	110

Theorem 2. *If the Sigma-protocols* EqCom, AccRel, ProdCom, *and* RangeProof *satisfy zero-knowledge ([BS20] Def. 19.5) and simulation extractability (A.1), and Strong-RSA holds in QR_n ([BP97]) then protocol* BListNM *satisfies soundness (Def. 3).*

Intuition. In Game 3 for soundness, the adversary $\mathcal{A}$ wins the game if he produces a proof, which, when extracted, gives a credential id, a, B, such that either the id belongs to the blocklist, i.e., $\mathsf{id} \in \mathcal{L}$ or id belongs to an honest UE, i,e. $\mathsf{id} \in \mathcal{H}$. The probability that $\mathcal{A}$ wins is the sum of the probabilities of winning in each case (the cases are independent).

To show that it is infeasible for any $\mathcal{A}$ to provide a proof where $\mathsf{id} \in \mathcal{L}$, we use the properties of the RSA accumulator. A revoked id which verifies is not a factor of the values $\{\mathsf{id}_1, \ldots, \mathsf{id}_n\} \in \mathcal{L}$ (so two values in the $\mathcal{L}$ cancel out) or is equal to $1 \mod \phi(n)$. Finding such an id would break the Strong-RSA assumption.

For any $\mathsf{id} \in \mathcal{H}$, $\mathcal{A}$ never learned a credential. By simulation extractability, even after seeing many proofs, the adversary cannot extract a (new) credential. The formal proof is at Appendix A.3. □

Non-replayability. By Theorem 2, even after observing many proofs from several UEs, an adversary still cannot craft any *new* accepting proof without a valid credential. Even if the adversary simply copies or performs a replay attack, it is still hard to pass the non-membership proof because each non-membership proof is associated with a unique timestamp tms (See Fig. 5) provided by the network. Each non-membership proof is tied to a specific tms. Thus, each proof is replayable only within a certain time window (related to the validity of each timestamp tms), which restricts the attack surface for an adversary. This improves the current non-anonymous proof mechanism that is trivially spoofable. Future work could investigate integrating techniques from distance bounding to further reduce or eliminate the vulnerability window.

Theorem 3. *If* SS *is unforgeable, then protocol* BListNM *satisfies revocation unforgeability (Def. 4).*

Intuition. We prove revocation unforgeability by reduction to unforgeability of the signature. The idea is that in our scheme, VerUpdBList is exactly SS.Ver. Thus, breaking revocation unforgeability immediately yields breaking unforgeability. The formal reduction is provided in the Appendix A.4. □

8 Evaluation

In this section we evaluate the performance of Enroll, Auth, Upd, UpdBList, and Vrfy functions. We consider the amount of time to run the functions and the sizes of the outputs.

Table 3: UE Time Taken to prove

Measure (ms)	Mean	Std Dev.
Onl. Auth Time	3432	77
Off. Auth Time	14019	52

Table 4: UE Credential Sizes (Bytes)

Name	Mean	Std. Dev.
Credentials ($\mathsf{sk}_{\mathsf{id}}$)	384	0.39
Identity (id)	194	0.41

Table 5: UE Time Taken to update credentials (ms)

Measure	# of Upd	Mean	Std Dev.
Upd. Time	1	756	8.45
	100	37896	284.23

8.1 Implementation

We build on an open source Python implementation of RSA accumulator[4]. We use their helper functions (e.g. prime number generation, hash-to-prime, etc.) and their function to create non-membership credentials to instantiate the Setup and Enroll functions. Note that this library only creates credentials and so their "proof" is not zero-knowledge. Our implementation extends their library with instantiations of Auth and Vrfy that allow a user to prove non-membership in zero-knowledge. More specifically, we implement the Sigma protocols defined in Appendix A. Finally, we instantiate the UpdCred function, which also contributes to the open-source library that currently does not have functions to update witnesses on deletions from the blocklist. There exist implementations of the range proof[CFT98] at [Ban], proof of equality of commitments[Bou00] and proof of product of commitments[DF02] at [XS] in Go. We anticipate that an implementation in a more performant language than Python would likely lead to significant performance improvements, so our results are a conservative lower bound on performance.

8.2 Experimental Setup

For our experiment testbed, the network operations (Enroll, UpdBList, Vrfy) and the UE operations (Auth, Upd) will execute on vastly different hardware classes. We thus measured the server-side calculations on a MacBook Pro with a 2.3GHz Dual-Core Intel Core i5 processor. The UE-side calculations were done with a Raspberry Pi Model B1 with an ARM11 CPU (which is used also used in cellular modems from Qualcomm and Broadcom). The Raspberry Pi serves as a reasonable, conservative, analogue for cellular modem chip capabilities. Finally, we use the MakerHawk USB Power Meter Tester as an ammeter. We place the MakerHawk between the power supply and Raspberry Pi. We run the UE operations and measure the average amperage.

In our experiments we run the Setup() function to set the parameters. We set our key sizes to be 3072 bits to arrive at 128-bit-equivalent security [BBB+06]. We then run tests with 1000 enrolled users and 100 revoked users. We measure the network-side protocols (Enroll, UpdBList, Vrfy) in Sec 8.3 and UE-side protocols (Auth, Upd) in Sec 8.4.

8.3 Network Side Protocols

Our network-side experiments measure times for user enrollment, verification of a user, and revocation of a user. We present the mean size of the credentials in Table 4 and the mean time taken to enroll a user (and thus to create the credentials) in Table 1.

Measuring Enroll**:** Enrollment requires a randomized hash-to-prime (H_{pr}) algorithm. H_{pr} is quick in most cases, with occasional larger outliers. For evaluation, we ran the full enrollment protocol 1000 times. We observed that the distribution of results was heavily skewed (skewness = 2.65) toward lower values, implying that in most cases the algorithm is fast. Because the distribution is skewed low, the geometric mean and standard deviation are the most appropriate measures of central tendency. We find that the geometric mean is 939 ms with a standard deviation of 2.65. We note that the enrollment is only a one-time cost that can be computed offline by the CEIR maintainer.

[4]https://github.com/oleiba/RSA-accumulator

Measuring Vrfy**:** Our next experiment has 1000 users create a proof that they are not in the blocklist. We run the verify algorithm on these 1000 proofs and measure the average time to verify a proof in Table 1. The main contributor here is the multi-exponentiation the verifier needs to do in each of the PoK. The number of times a Vrfy operation is run depends on the number of times a device attaches to the network. Moreover, the AMF need not request a PEI proof on every attach. Therefore the number of times the network needs to run $\mathsf{Vrfy}()$ depends only on an attach for which the PEI is requested, thus not universally affecting the network attach process.

Measuring UpdBList**:** Our next experiment is to revoke 100 users and measure the average time to add a user to the blocklist as well as the size of the update messages for each revocation update. Table 1 shows the mean time taken to revoke a user. The average size of an update message is 524 bytes with a standard deviation of 9.29.

The CEIR maintainer runs a revoke operation. First, it computes the updated blocklist using the revoked user's id and signs the list and the id to authenticate itself. It then sends the revoked user's id, the updated blocklist and the signature to the user. We note that the number of revocations depends on the reported devices for blocklisting.

8.4 UE-Side Protocols

For the UE side experiments we measure time taken to compute a proof and time taken to update credentials. We also measure the size of the credentials $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ that are stored in the device. In practice, the network only requests the PEI be authenticated when the device is authenticating with a new AMF and not necessarily at every tower.

Measuring Auth**:** First, we run Auth for all 1000 enrolled parties. We measure the average time taken to compute a proof in Table 5. We measure the online and the offline part of the proof separately. The UE computes the commitments of the secret credentials offline. Some messages of the sigma protocol include a timestamp (tms). The UE must calculate these when attaching to a network in an online phase. By dividing proof times into an offline and online phase we cut down on the proof creation time when a UE attaches to the network by 80%. When creating the proofs, the first round of messages of the Sigma protocol can be pre-computed as well. Thus, offline time is the commitment time plus the time for the first step of the proof, and online time is the remaining two steps of each sigma protocol. The total proof size is 10462 bytes on average (summing over the offline and online parts). Times are in Table 3.

Measuring Upd**:** We revoke 100 parties and have the remaining 900 parties update their credentials. We measure the time taken to update these credentials in Table 5. A user receives all the update messages for the day (or some other regular interval) and updates its credentials. The user will update its credentials for each revoked update message. In our implementation, updates are offline and need not happen during the attach process with the network, so the latency of the attach protocol is not affected by the update process.

Finally, in Table 4 we measure the size of the credentials that are stored in the device. The ID is stored in the UE and takes up 194 bytes of storage. The credentials take only about 384 bytes of space.

Battery Usage: We run Auth for the 100 enrolled parties and measure the current. Similarly, we will measure Upd to see how much amperage is taken. The Pi takes in 5.11V (on average).

First, we run 100 authentications. Second, we add 5 PEIs to the list and then have the remaining 95 parties run Upd. The current drawn is similar to the authentication. We multiply the average current drawn over time to get the overall charge used. An average phone may have 2500-4000 mA·hr of charge, so authentication takes 0.006055% of the battery at most. The values are in Table 2.

8.5 Embedding to the current 5G architecture

The UE needs to update its credentials with each update to the blocklist. Of course, we cannot have the UE update its credentials only when the network requests the PEI. This is inefficient and would be detrimental to the quality-of-service. Instead, we envision a mechanism where the UE can download the latest updates to the blocklist at regular intervals (e.g. everyday) and update its credentials. The UE can also do some pre-computation for its proof and therefore the actual online time to create a proof will be small.

Next we outline the different messages that include PEI when the network requests it in the current implementation [3rda] and address what changes one would need to make to incorporate our protocols.

Security Mode Complete: At the end of the AKA protocol, the network asks for the PEI when it initiates a Security Mode Procedure message. The UE responds with a Security Mode Complete command which includes the PEI. Note that these messages are encrypted and integrity protected between the UE and the AMF (home network). In our setting, the Security Mode Procedure message will include an updated accumulator and recently blocklisted PEIs and the Security Mode Complete will include the proof that the UE's PEI is not in the blocklist.

Identity Response: The Identification Procedure allows the AMF to request the UE for specific identification parameters that include the PEI. The UE sends the response, which includes the PEI in an IDENTITY RESPONSE message. These messages are encrypted and integrity protected as well between the UE and the home network. Like the Security Mode Command above, the UE will send a proof instead of the PEI in the IDENTITY RESPONSE message and the network will need to send the updated blocklist as part of the IDENTITY REQUEST message.

Emergency Registration: This is the only case when the PEI is sent unencrypted. When the SUPI is not available the UE registers for emergency services with a PEI. The registration process is like the initial registration by a UE to the network, except the PEI need not be authenticated and checks for access restrictions are not done. Thus, the UE could provide a random value for the PEI and register for the emergency services.

Other architectural changes: In our protocols, when the AMF needs to authenticate a user, it needs the most updated accumulator. To ensure this, the AMF will need to query the 5G-EIR for the updated blocklist each time it requests the PEI from a UE. This incurs an extra round of communication between the 5G-EIR and the AMF. Finally, the PEI is also part of the "context" in many network flows. For example, when a UE switches from one AMF to another, the old AMF sends the PEI along with other identifiers to the new AMF as part of a security context. This security context is useful in the hand-off. In our setting we conjecture this can be done by using the proof instead of the PEI in the security context.

9 Related Work

In this section, we describe work in cellular network security relevant to our problem. For a detailed treatment, we refer the reader to a recent survey from Rupprecht et al. [RDH+18].

Most work on cellular network identifier security has focused only on IMSI catchers, devices that passively listen or pose as a legitimate base station to enable tracking of users or interception of communications. IMSI catchers are capable of also collecting and using the equipment identifier. Prior work has focused on detecting active tracking and interception attacks [DPK+14, DPW16, NSCK17, LWW+17, PSB+17], or improving cellular protocols to prevent successful attacks [KM15, vVd15, AMR+12, SHC+20]. In cellular networks, a temporary mobile subscriber identifier (TMSI) is also used to prevent leakage of the permanent subscriber identifier. Prior work has found that networks fail to manage these securely [DJNY12, HBK18], and even if perfectly managed attackers can still exploit side channels to attack users [RKHP19, HEC+19]. These kinds of attacks can be used to leak location by listening to the paging channel and stress testing the network. Our work contributes to the growing body of literature on 5G security, which has used formal methods to audit the new security protocols [HEK+19, BDH+18, CD19] and proposed changes to improve security [KDM18].

Our work contributes to the growing literature on anonymous telecommunication. Hauser et al. developed a system called Phonion to enable creation of temporary, anonymous telephone numbers for untraceable anonymous phone calls [SBP+17], but this system does not protect the identity of users from their own network, merely the anonymous calling or called party. Camenisch et al. describe a protocol that uses anonymous credentials to allow a phone a mobile network to collaborate to prove to a third party that a specific phone is in an area without revealing the specific location as a secondary authentication factor [COP15]; this protocol also provides no anonymity between the mobile user and the network.

Finally, we discuss related cryptographic constructions. Benaloh and De Mare [BDM93] introduced the concept of an accumulator to store a set of values. Our protocol uses the RSA-based accumulators that allows one to give zero-knowledge non-membership proofs by [LLX07, CL02]. Ghosh et al. [GOP+16] present zero-

knowledge accumulators with a different privacy goal, where they show the protocol execution of accumulator related operations leaks nothing to a client or an external adversary. Baldimtsi et al. [BCD+17] also propose an RSA-based accumulator and present an anonymous revocation component.

10 Conclusion

Historically, mobile networks have required strong positive identity checks to protect the network from abuse, but these checks came at the cost of trusting the network with the ability to monitor and track a user's physical location and network activities. In this work we presented a protocol that enables cellular devices to anonymously prove that they are authorized to access the network by virtue of not being listed in the CEIR blocklist. Our findings are that with the help of cryptographic primitives like zero-knowledge proofs and RSA accumulators we can have practical anonymous device authorization. Our work can form the basis of future mobile network designs that provide strong, provable privacy guarantees.

References

[3rda] 3rd Generation Partnership Project. 3GPP TS 24.501 V16.5.1 . Technical Report Non-Access-Stratum (NAS) protocol for 5G System. https://www.etsi.org/deliver/etsi_ts/124500_124599/124501/16.05.01_60/ts_124501v160501p.pdf.

[3rdb] 3rd Generation Partnership Project. 3GPP TS 29.511 V15.6.0. Technical Report Equipment Identity Register Services. http://www.3gpp.org/ftp//Specs/archive/29_series/29.511/29511-f60.zip.

[3rdc] 3rd Generation Partnership Project. 3GPP TS 33.501 V15.8.0 . Technical Report Security architecture and procedures for 5G system. http://www.3gpp.org/ftp//Specs/archive/33_series/33.501/33501-f80.zip.

[AMR+12] Myrto Arapinis, Loretta Mancini, Eike Ritter, Mark Ryan, Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. New Privacy Issues in Mobile Telephony: Fix and Verification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, New York, NY, USA, 2012. ACM.

[Ban] ING Bank. ing-bank/zkrp.

[BBB+06] Elaine Barker, William Barker, William Burr, William Polk, Miles Smid, et al. *Recommendation for key management: Part 1: General*. National Institute of Standards and Technology, Technology Administration, 2006.

[BCD+17] Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. Accumulators with applications to anonymity-preserving revocation. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 301–315. IEEE, 2017.

[BDH+18] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A Formal Analysis of 5G Authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, New York, NY, USA, 2018. ACM.

[BDM93] Josh Benaloh and Michael De Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 274–285. Springer, 1993.

[Béz79] Etienne Bézout. *Théorie générale des équations algébriques par M. Bézout*. de l'imprimerie de Ph.-D. Pierres, rue S. Jacques, 1779.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 431–444. Springer, 2000.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *International conference on the theory and applications of cryptographic techniques*, pages 480–494. Springer, 1997.

[BS20] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.5*, 2020.

[CD19] Cas Cremers and Martin Dehnel-Wild. Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion. In *Proceedings 2019 Network and Distributed System Security Symposium*, San Diego, CA, 2019. Internet Society.

[CFT98] Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come—easy go divisible cash. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 561–575. Springer, 1998.

[CH10] Philippe Camacho and Alejandro Hevia. On the impossibility of batch update for cryptographic accumulators. In *International Conference on Cryptology and Information Security in Latin America*, pages 178–188. Springer, 2010.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Annual International Cryptology Conference*, pages 61–76. Springer, 2002.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 402–414. Springer, 1999.

[COP15] Jan Camenisch, Diego Alejandro Ortiz-Yepes, and Franz-Stefan Preiss. Strengthening Authentication with Privacy-Preserving Location Verification of Mobile Phones. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society - WPES '15*, Denver, Colorado, USA, 2015. ACM Press.

[CS00] Ronald Cramer and Victor Shoup. Signature schemes based on the strong rsa assumption. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):161–185, 2000.

[DF01] Ivan Damgard and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. Cryptology ePrint Archive, Report 2001/064, 2001. `http://eprint.iacr.org/2001/064`.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 125–142. Springer, 2002.

[DJNY12] Denis Foo Kune, John Koelndorfer, Nicholas Hopper, and Yongdae Kim. Location leaks over the GSM air interface. In *NDSS 2012*, February 2012.

[DPK+14] Adrian Dabrowski, Nicola Pianta, Thomas Klepp, Martin Mulazzani, and Edgar Weippl. IMSI-catch Me if You Can: IMSI-catcher-catchers. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, New York, NY, USA, 2014. ACM.

[DPW16] Adrian Dabrowski, Georg Petzl, and Edgar R. Weippl. The Messenger Shoots Back: Network Operator Based IMSI Catcher Detection. In *Research in Attacks, Intrusions, and Defenses*. Springer, Cham, September 2016.

[DT08] Ivan Damgård and Nikos Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. *IACR Cryptol. ePrint Arch.*, 2008:538, 2008.

[FCC20] FCC Proposes Over $200M in Fines for Wireless Location Data Violations, February 2020.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[FT14] Pierre-Alain Fouque and Mehdi Tibouchi. Close to uniform prime number generation with fewer random bits. In *International Colloquium on Automata, Languages, and Programming*, pages 991–1002. Springer, 2014.

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 123–139. Springer, 1999.

[GM17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. pages 581–612, 2017.

[GOP+16] Esha Ghosh, Olga Ohrimenko, Dimitrios Papadopoulos, Roberto Tamassia, and Nikos Triandopoulos. Zero-knowledge accumulators and set algebra. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 67–100. Springer, 2016.

[GSM19] The state of mobile internet connectivity 2019, July 2019.

[GZJS12] Michael C Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 101–112, 2012.

[Han19] Lucy Handley. Nearly three quarters of the world will use just their smartphones to access the internet by 2025. 4, 1 2019.

[HBK18] Byeongdo Hong, Sangwook Bae, and Yongdae Kim. GUTI Reallocation Demystified: Cellular Location Tracking with Changing Temporary Identifier. In *Proceedings 2018 Network and Distributed System Security Symposium*, San Diego, CA, 2018. Internet Society.

[HEC+19] Syed Rafiul Hussain, Mitziu Echeverria, Omar Chowdhury, Ninghui Li, and Elisa Bertino. Privacy Attacks to the 4G and 5G Cellular Paging Protocols Using Side Channel Information. In *Proceedings 2019 Network and Distributed System Security Symposium*, San Diego, CA, 2019. Internet Society.

[HEK+19] Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. 5GReasoner: A Property-Directed Security and Privacy Analysis Framework for 5G Cellular Network Protocol. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London United Kingdom, November 2019. ACM.

[KDM18] Haibat Khan, Benjamin Dowling, and Keith M. Martin. Identity Confidentiality in 5G Mobile Telephony Systems. In Cas Cremers and Anja Lehmann, editors, *Security Standardisation Research*, Lecture Notes in Computer Science, Cham, 2018. Springer International Publishing.

[KM15] Mohammed Shafiul Alam Khan and Chris J. Mitchell. Improving Air Interface User Privacy in Mobile Telephony. *arXiv:1504.03287 [cs]*, April 2015.

[Lin17] Yehuda Lindell. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. Springer, 2017.

[LLX07] Jiangtao Li, Ninghui Li, and Rui Xue. Universal accumulators with efficient nonmembership proofs. In *International Conference on Applied Cryptography and Network Security*, pages 253–269. Springer, 2007.

[LWW+17] Zhenhua Li, Weiwei Wang, Christo Wilson, Jian Chen, Chen Qian, Taeho Jung, Lan Zhang, Kebin Liu, Xiangyang Li, and Yunhao Liu. FBS-Radar: Uncovering Fake Base Stations at Scale in the Wild. In *Proceedings 2017 Network and Distributed System Security Symposium*, San Diego, CA, 2017. Internet Society.

[Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. pages 369–378, 1988.

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. pages 275–292, 2005.

[NSCK17] Peter Ney, Ian Smith, Gabriel Cadamuro, and Tadayoshi Kohno. SeaGlass: Enabling City-Wide IMSI-Catcher Detection. *Proceedings on Privacy Enhancing Technologies*, 2017(3), July 2017.

[PAS+18] Christian Peeters, Hadi Abdullah, Nolen Scaife, Jasmine Bowers, Patrick Traynor, Bradley Reaves, and Kevin Butler. Sonar: Detecting ss7 redirection attacks with audio-based distance bounding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 567–582. IEEE, 2018.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 387–398. Springer, 1996.

[PSB+17] Shinjo Park, Altaf Shaik, Ravishankar Borgaonkar, Andrew Martin, and Jean-Pierre Seifert. White-Stingray: Evaluating IMSI Catchers Detection Applications. In *USENIX Workshop on Offensive Technologies (WOOT)*, 2017.

[RDH+18] David Rupprecht, Adrian Dabrowski, Thorsten Holz, Edgar Weippl, and Christina Pöpper. On Security Research Towards Future Mobile Network Generations. *IEEE Communications Surveys Tutorials*, 20(3), thirdquarter 2018.

[RKHP19] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Popper. Breaking LTE on Layer Two. In *2019 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, May 2019. IEEE.

[SBP+17] Stephan Heuser, Bradley Reaves, Praveen Kumar Pendyala, Henry Carter, Alexandra Dmitrienko, William Enck, Negar Kiyavash, Ahmad-Reza Sadeghi, and Patrick Traynor. Phonion: Practical Protection of Metadata in Telephony Networks. *Proceedings of Privacy Enhancing Technologies*, 2017(1), July 2017.

[SHC+20] Ankush Singla, Syed Rafiul Hussain, Omar Chowdhury, Elisa Bertino, and Ninghui Li. Protecting the 4G and 5G Cellular Paging Protocols against Security and Privacy Attacks. *Proceedings on Privacy Enhancing Technologies*, 2020(1), January 2020.

[VD07] Vassilis Prevelakis and Diomidis Spinellis. The Athens Affair. *IEEE Spectrum*, June 2007.

[vVd15] Fabian van den Broek, Roel Verdult, and Joeri de Ruiter. Defeating IMSI Catchers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, Denver, Colorado, USA, 2015. ACM Press.

[XS] Xlab-Si. xlab-si/emmy.

$\mathsf{ProveAccRel}(\mathsf{X} = (c_e, c_a, c_z, c_B, \mathfrak{g}, \mathfrak{h}, n, \mathsf{stmt}),$ $\mathsf{VerAccRel}(\mathsf{X} = (c_e, c_a, c_z, c_B, \mathfrak{g}, \mathfrak{h}, n, \mathsf{stmt}),$
$\mathsf{W} = (\mathsf{id}, r_e, a, r_a, z, r_z))$ $\mathsf{ch}_2, E_1, E_2, E_3, E_4, E_5, E_6)$

Note: $c_e = (c_B)^{\mathsf{id}}\mathfrak{h}^{r_e}, z = w \cdot \mathsf{id}$

Parse: $\mathsf{stmt} = (\mathsf{list\text{-}pk}, \mathsf{tms})$

$\sigma, \tau, \varepsilon, \rho_a, \rho_z, \rho_e \leftarrow_{\$} \{0,1\}^{|n|}$

$W_1 = \mathsf{list\text{-}pk}^{\sigma}\mathfrak{g}^{\tau}\mathfrak{h}^{\rho_e}, W_2 = \mathfrak{g}^{\sigma}\mathfrak{h}^{\rho_a}$

$W_3 = \mathfrak{g}^{\tau}\mathfrak{h}^{\rho_z}, W_4 = (c_B)^{\varepsilon}h^{\rho_e}$

$\mathsf{ch}_2 = H(W_1 \| W_2 \| W_3 \| W_4 \| \mathsf{X})$

$E_1 = \sigma + \mathsf{ch}_2 a, E_2 = \tau + \mathsf{ch}_2 z$

$E_3 = \varepsilon + \mathsf{ch}_2 \mathsf{id}, E_4 = \rho_a + \mathsf{ch}_2 r_a$

$E_5 = \rho_z + \mathsf{ch}_2 r_z, E_6 = \rho_e + \mathsf{ch}_2 r_e$

Output $(\mathsf{ch}_2, E_1, E_2, E_3, E_4, E_5, E_6)$

Check $\mathsf{ch}_2 = H(\mathsf{list\text{-}pk}^{E_1} g^{E_2} h^{E_6} ((c_e g)^{-\mathsf{ch}_2} \|$
$\| g^{E_1} h^{E_4} c_a^{-\mathsf{ch}_2} \| g^{E_2} h^{E_5} c_z^{-\mathsf{ch}_2} \| c_B^{E_3} h^{E_6} c_e^{-\mathsf{ch}_2} \| \mathsf{X}))$

Figure 11: Proving Relation between committed values and $\mathsf{list\text{-}pk}_{\mathsf{id}}$

A Sigma-Protocols

In this section, we describe the Sigma-protocols that are executed by UE when computing a proof as shown in Fig. 6.

EqCom. Protocol EqCom is used to prove that two commitments C_1 and C_2 are commitments to the same value under different bases. We follow the Sigma-protocol from [Bou00]. EqCom is used to prove that c_{id} and c_e are both commitments of the secret id under bases $\mathfrak{g}, \mathfrak{h}$ and (c_B, h).

AccRel. Protocol AccRel in Fig. 11 is used to prove that the values committed in c_e, c_a, c_z, c_B and list-pk satisfy certain relations. The prover here wants to prove that the id is not in the accumulator, which means that the relationship $\mathsf{list\text{-}pk}^a = B^{\mathsf{id}}g$ holds. Since a, B, and id are hidden in the exponents of c_a, c_B, c_e, to prove the relation the prover will prove several connections about commitments.

ProdCom. Protocol ProdCom (from [DF02]) is used to prove that given 3 commitments C_3, C_1, C_2, the secret z committed in C_3 (where $C_3 = \mathfrak{g}^z\mathfrak{h}^{r_z}$) is the product $x \cdot y$ of the secrets committed in C_1 and C_2. The prover blinds the id using $z = \mathsf{id} \cdot w$ and must here prove that c_z is a commitment to the values committed to in c_w and c_{id}.

RangeProof. Protocol RangeProof (from [CFT98]) is used to prove that a committed value lies in a specific range. Otherwise, it would be possible to find an id that is not a single prime number, but a composite.

Full Security Proofs

A.1 Useful Definitions and Lemmata

A Sigma-protocol for a relation $\mathcal{R}$ has the properties of *simulation-extractability* and *special honest verifier zero-knowledge* (SHVZK).

Simulation extractability. This property typically requires that even after seeing many simulated proofs, if an adversary $\mathcal{A}$ produces a proof, it is possible to extract a witness from it. In this paper, we consider a stronger definition where we allow an adversary to also see statements and witnesses of the simulated proofs. Thus, producing a proof is trivial for an adversary. In this stronger definition, we say that an adversary breaks simulation extractability if she can produce a proof that when extracted, it opens to a *new* statement and witness. Our definition is based on definition 2.10 from Maller and Groth [GM17].

Special honest verifier zero-knowledge (SHVZK). This property requires that for a zero knowledge protocol we are able to define a simulator such that it can output a transcript that is indistinguishable from the real transcript without any access to the witness. For more details refer to Sec 19.1.1 of [BS20].

Definition 5 (Bézout's Identity)**.** If $s, x \in \mathbb{Z}$, $s, x \neq 0$, then there exists a, b such that $gcd(s, x) = as + bx$.

Lemma 1. Let $s, x \in \mathbb{Z}^+$. Let $(a, b) \leftarrow \mathsf{Béz}(s, x)$ so that $d = as+bx$. Then by the Definition 5, $d = \gcd(x, s)$. Furthermore, d is the minimum value that can be written as such a linear combination, i.e. $d = \min_{d^*}\{as+bx = d^*|d^* \in \mathbb{Z}\}$. Also, all values $d^* \in \mathbb{Z}$ that are linear combinations of s, x are multiples of d.

LR-hiding [Lin17] (Informal). Here, the adversary has access to an oracle which takes queries of two inputs – a left input and a right input. The oracle either returns a commitment on either the left input or on the right input in response to every query from the adversary. In the end, the adversary must decide which of the oracles (the one that outputs left input commitments or the one that outputs right input commitments) it was instantiated with.

A.2 Proof for Anonymity: Theorem 1

Proof. To break anonymity the adversary $\mathcal{A}$ needs to guess b as shown in the $\mathsf{GameAnon}^{\Pi}_{\mathcal{A}}(\lambda)$ (Game 2). Our strategy to prove that the advantage of $\mathcal{A}$ is negligible is to show through a series of hybrids that the transcript of the protocol when $b = 0$ is indistinguishable from the case when $b = 1$.

Proof by hybrids

- Hyb_0: Execution of Auth when $b = 0$. The transcript is the commitments $c_{\mathsf{id}} = \mathsf{Com}(\mathsf{id}_0)$, $c_a = \mathsf{Com}(a_0), c_B = \mathsf{Com}(B_0), c_e = \mathsf{Com}(\mathsf{id}_0)$ to base c_B, $c_w = \mathsf{Com}(w)$, $c_z = \mathsf{Com}(z_0)$ where $z = \mathsf{id}_0 \cdot w$, and $\pi_0 = \mathsf{PoK}_1 \ldots \mathsf{PoK}_5$.
- Hyb_1 is the same as Hyb_0 except that we replace π_0 with simulated zero-knowledge proofs. We prove in Lemma 2 that Hyb_1 is indistinguishable from Hyb_0 due to the zero-knowledge propery of the sigma protocols.
- Hyb_2 is the same as Hyb_1 except that we replace c_e with commitment to id_1. We prove in Lemma 3 that Hyb_1 and Hyb_2 are indistinguishable because of the LR-hiding property of the commitment scheme.
- Hyb_3 is the same as Hyb_2 except that we replace $c_B = B_0 \cdot \mathfrak{g}^w$ with $B_0 \cdot \mathfrak{g}^w$. Since w is randomly sampled, $B_0 \cdot \mathfrak{g}^w$ has a uniform distribution over $\mathbb{Z}_n$. Likewise, $B_1 \cdot \mathfrak{g}^w$ has a uniform distribution over $\mathbb{Z}_n$. Thus, we show that Hyb_3 is indistinguishable from Hyb_2.
- Hyb_4 is the same as Hyb_3 except that we replace commitments $c_a, c_{\mathsf{id}}, c_z$ with commitments to a_1, id_1 and $z_1 = \mathsf{id}_1 \cdot w$. We prove in Lemma 4 that Hyb_3 is indistinguishable from Hyb_4 due to the LR-hiding property of the commitment scheme.
- Hyb_5 is the same as Hyb_4 except that we replace simulated zero-knowledge proofs with $\pi_1 = \mathsf{PoK}_1 \ldots \mathsf{PoK}_5$ which are the 5 sigma protocols in the case that bit 1 was chosen. Hyb_5 is the same as Hyb_4 are indistinguishable for the same reason Hyb_0 and Hyb_1 are indistinguishable.

Note that Hyb_0 is the case when $b = 0$ is picked by the challenger of the anonymity game, and Hyb_5 is the case when $b = 1$ is picked by the challenger of the anonymity game. Since Hyb_0 is indistinguishable from Hyb_5 we prove that the advantage of $\mathcal{A}$ in $\mathsf{GameAnon}^{\Pi}_{\mathcal{A}}(\lambda)$ is negligible.

□

Lemma 2. If the $\mathsf{EqCom}, \mathsf{AccRel}, \mathsf{ProdCom}$, and $\mathsf{RangeProof}$ satisfy zero-knowledge, then Hyb_1 is indistinguishable from Hyb_0.

Proof. Each of the PoK_i for $i \in [1, 5]$ have a simulator that computes a transcript indistinguishable from the real transcript. The technique to make the simulator is the same for every PoK_i: the simulator chooses the challenge at random, calculates the response knowing the challenge ahead of time, and reprograms the random oracle to output the challenge based on the response. Since all values are chosen randomly in the real world and the ideal world, the real transcript and the simulated transcript are distributed identically.

Since the proof of knowledge in the proving algorithm is an AND composition of each of these smaller proofs of knowledge, we can say that the simulated proof of knowledge and the real world proof of knowledge are indistinguishable. Since each simulated transcript is indistinguishable from the real transcript, we conclude that Hyb_0 is indistinguishable from Hyb_1.

□

Lemma 3. If the Fujisaki-Okamoto scheme satisfies LR-hiding ([Lin17] Chap. 6.5.2), then Hyb_2 is indistinguishable from Hyb_1.

Proof. Suppose there exists $\mathcal{A}$ that wins Hyb_0 and Hyb_1 with non-neglible difference. We construct a reduction $\mathcal{B}$ that wins the $\mathsf{LRHiding}$ game. $\mathcal{B}$ responds to $\mathcal{A}$'s queries as follows:

- $\mathsf{Corrupt}(\mathsf{id})$: $\mathcal{B}$ calculates Bezout coefficients using r.
- $\mathsf{Auth}(\mathsf{list\text{-}pk}, \mathsf{id}, \cdot)$: $\mathcal{B}$ computes $(a, B) \leftarrow \mathsf{Béz}(\mathsf{list\text{-}pk}, \mathsf{id})$ and creates proof normally.
- $\mathsf{UpdBList}(\mathsf{list\text{-}pk}, \mathcal{L}, \mathsf{id}, \cdot)$: output $(\mathcal{L}^*, \mathsf{list\text{-}pk}^*, \mathsf{upd})$ where $\mathsf{list\text{-}pk}^* = \mathsf{list\text{-}pk}^{\mathsf{id}}$, $\mathcal{L}^* = \mathcal{L} \cup \{\mathsf{id}\}$ and $\mathsf{upd} = (\mathsf{id}, \mathsf{SS.Sign}(\mathsf{id}))$

At the end $\mathcal{B}$ will output the same as $\mathcal{A}$. Now, if $\mathcal{A}$ wins in Hyb_0 and Hyb_1 with non-negligible difference, there exists a distinguisher D that distinguishes between Hyb_1 and Hyb_2 with probability greater than negligible. That is,

$$Pr[D(\mathsf{Hyb}_2) = 1] - Pr[D(\mathsf{Hyb}_1) = 1] > \mathsf{negl}(n)$$

Observe that when $b = 1$ in the LR-oracle experiment, the commitments C_e are generated as if in Hyb_2. This implies :

$$Pr[\mathsf{LRHiding}(1^n) = 1 | b = 1] = Pr[D(\mathsf{Hyb}_2) = 1]$$

Similarily, when $b = 0$ in the LR-oracle experiment, the commitments C_e are generated as if in Hyb_1. This implies :

$$Pr[\mathsf{LRHiding}(1^n) = 1 | b = 0] = Pr[D(\mathsf{Hyb}_1) = 0]$$

If the distinguisher can distinguish the hybrids, it can win the LR-hiding game. This contradicts security of the commitment scheme, since we know that Fujisaki-Okamoto scheme is LR-hiding. Thus, no adversary can distinguish between Hyb_1 and Hyb_2 □

Lemma 4. If the FO commitment satisfies LR-hiding ([Lin17] Chap. 6.5.2), then Hyb_4 is indistinguishable from Hyb_3

Proof. Similar to the proof of Lemma 3. □

A.3 Proof for Soundness: Theorem 2

Proof. In the authentication soundness game in Game 2, the adversary wins if he produces a proof which, when extracted, gives id, a, B for which either the id belongs to the blocklist, i.e., $\mathsf{id} \in \mathcal{L}$ or id belongs to an honest UE, i,e. $\mathsf{id} \in \mathcal{H}$. We show that in the authentication soundness game instantiated with protocol BListNM, the adversary can produce such an id with only negligible probability, so the ability to break authentication soundness is negligible as well. The probability that the adversary wins is the sum of the probabilities of winning in each case.

Case 1: $\mathsf{id} \in \mathcal{L}$. It is infeasible for an adversary to provide us with a proof that, when extracted, gives $\mathsf{id} \in \mathcal{L}$. A revoked id that verifies either is not a factor of the values $\{ \mathsf{id}_1, \ldots, \mathsf{id}_n \}$, meaning that two values in $\mathcal{L}$ cancel other out, or the adversary has found a value id which is equivalent to $1 \mod \phi(n)$ where n is the RSA-modulus. Finding such an id breaks the Strong-RSA assumption.

For the reduction, assume for sake of contradiction that an adversary $\mathcal{A}$ to soundness exists. Then we can construct an adversary $\mathcal{B}$ to Strong-RSA. The Proving Algorithm used in BListNM is an AND-composition of the PoK, thus an adversary that creates a valid proof must have been able to create valid proof for each of the PoK. With knowledge soundness, it is possible for a reduction to extract the witnesses to an adversary's proof, which will then allow a reduction to break Strong-RSA by leveraging the properties from the Bézout coefficients (Def. 5) and Lemma 1.

Following the proving algorithm in Fig. 6, the witness consists the witness consists of $(\mathsf{id}, r_e, r_{\mathsf{id}}, a, r_a, z, r_z, w, r_w, r_{\mathsf{id}}, a + 2^\ell, r_a)$. Then $\mathcal{B}$ can parse id and a from the witness and learn B by calculating $c_B / \mathfrak{g}^w$. Now, a revoked id^* can only verify if $\mathsf{id}^* = 1 \mod \phi(n)$ or if there exists another $\mathsf{id} \in \mathcal{L}$ such that $id\mathsf{id}^* = 1 \mod \phi(n)$. The former case is already excluded because the output id is required to be prime. For the latter, $\mathcal{B}$ can go through the list of id's and check which pair $\mathsf{id}, \mathsf{id}^*$ have that $y^{\mathsf{id}} = C$ and $C^{\mathsf{id}^*} = y$ and produce y, id as her answer, thus breaking Strong-RSA. $\mathcal{B}$ can win with the same probability as $\mathcal{A}$. By the hardness of strong-RSA, $\mathcal{B}$ has at most a negligible probability, so $\mathcal{A}$ can win with no better than negligible probability as well.

Case 2: $\mathsf{id} \in \mathcal{H}$. This is an id for which the adversary never learned a credential. The concept of simulation extractability captures that even after seeing many proofs, the adversary should not be able to extract a (new) credential. We need to take two steps (by hybrid argument). The reason the hybrid step is necessary is we do not have a single assumption or theorem from which we can reduce the soundness game, as $\mathcal{A}$ receives honestly generated proofs as well as witnesses. By simulation extractability (Def. 5) one can only prove instances to which one knows a witness, even after seeing many simulated proofs. Special-sound interactive protocols made non-interactive by using the Fiat Shamir transform are simulation extractable. Each of EqCom, AccRel, ProdCom, and RangeProof are such interactive protocols. We also use the ZK property to ensure that such an adversary cannot distinguish between real proofs and simulated proofs. This allows a reduction to use an adversary to authentication soundness, giving it many simulated proofs, and in the end, the reduction can retrieve a witness.

To prove that no adversary can produce an $\mathsf{id} \in \mathcal{H}$, we morph from allowing the adversary to see honest proofs to simulated proofs, and then use this new game to show a reduction to simulation extractability.

Hyb_0 to Hyb_1: Hyb_0 is the same as the authentication soundness game. Hyb_1 is the same except for on Authenticate queries, the adversary receives a simulated proof instead. The ZK property means that for every PPT verifier, V, there is a PPT algorithm ZK.Sim such that the output of ZK.Sim is indistinguishable from an honest proof. ZK.Sim itself does not use the witness, so the verifier cannot learn anything about the witness from the output. Honest proofs must also therefore leak no information. To break the ZK property, an adversary must be able to distinguish whether, upon requests for (polynomially many) proofs, it is receiving proofs from a simulator or from running the proof system between the prover and verifier.

In conclusion, as an adversary can produce an $\mathsf{id} \in \mathcal{H}$ with no better than non-negligible probability (due to the zero-knowledge and simulation-extractability properties) and an $\mathsf{id} \in \mathcal{L}$ with no better than negligible probability (due to the Strong-RSA) property, no adversary can produce an $\mathsf{id} \in \mathcal{H} \cup \mathcal{L}$ except with negligible probability as well. Soundness holds.

Hyb_1 to simulation extractability. Next, assuming there is an adversary who can win in the game Hyb_1 with some probability, we construct an adversary $\mathcal{B}_{se}$ for simulation extractability. Following the definition in 5, $\mathcal{B}_{se}$ has access to oracle $\mathcal{O}$ which allows for proofs from $\mathsf{NIZK}.Sim$ and access to the oracle $\mathcal{C}$ which allows $\mathcal{B}$ to see statements and witnesses.

If $\mathcal{A}$ wins, it means that this proof verifies and that $\mathcal{A}$ never queried for a proof on this list-pk, id, and never queried for a witness on this id. By the post-conditions for simulation-extractability, (id, a, B) is not a valid witness for list-pk with only negligible probability. Thus $\mathsf{id} \in \mathcal{H}$ with high probability and $\mathcal{B}_{se}$ can extract a witness for a new id with high probability relative to the probability that $\mathcal{A}$ produces a valid proof. However, the probability that $\mathcal{A}_{se}$ should be able to produce such a valid witness is negligible (due to simulation extractability). Thus, $\mathcal{A}$ can win in Hyb_1 with only negligible probability as well. Taken together with the fact that the difference between an adversary winning in Hyb0 and Hyb_1 is negligible as well, we conclude that any adversary $\mathcal{A}$ can win authentication soundness with at most negligible probability and thus BListNM satisfies authentication soundness.

By assumption, it is infeasible for any PPT adversary to break simulation-extractability and each of the steps we took are indistinguishable from the last, meaning that it is infeasible for any adversary to break authentication soundness as well. □

A.4 Proof of Revocation Unforgeability: Theorem 3

Proof. Assume there exists an adversary $\mathcal{A}$ who wins revocation unforgeability, i.e. that $Pr[\mathsf{GameRevForge}^{\Pi}_{\mathcal{A}}(\lambda) = 1] = p(n)$ where p is non-negligible function p. We construct an adversary $\mathcal{B}$ for the signature scheme SS that uses $\mathcal{A}$ to attack unforgeability. After activating $\mathcal{A}$, $\mathcal{B}$ on all oracle queries from $\mathcal{A}$, responds faithfully as the challenger would. On queries to UpdBList on id, $\mathcal{L}$, $\mathcal{B}$ will request a signature $\mathsf{sigUpmsg} \leftarrow \mathsf{SS.Sign}(\mathsf{list\text{-}pk}, \mathsf{id})$ from its challenger and return $\mathsf{upd} = ((\mathsf{list\text{-}pk}, \mathsf{id}), \mathsf{sigUpmsg})$ to $\mathcal{A}$. When $\mathcal{A}$ returns $(\mathcal{L}, \mathsf{list\text{-}pk}^*, \mathsf{upd}^*)$, $\mathcal{B}$ sets as her forgery: $(\mathsf{list\text{-}pk}^*, \mathsf{id}^*)$ and upd^*.

If $\mathcal{A}$ wins, $\mathcal{B}$ wins with the same probability. First, it must be the case that $(\cdot, \mathcal{L}^*, \mathsf{list\text{-}pk}^*, \mathsf{upd}^*) \notin \mathcal{L}$ already, so $\mathcal{A}$ cannot have asked for this particular id to have been revoked before. The next check is that $\mathsf{VrfyUpdBList}(\mathsf{mpk}, \mathsf{list\text{-}pk}, \mathsf{id}, \mathsf{sigUpmsg})$. By definition of BListNM, $1 \leftarrow \mathsf{SS.Vrfy}(\mathsf{mpk}, \mathsf{list\text{-}pk}, \mathsf{id}, \mathsf{sigUpmsg})$.

This means that $\mathcal{B}$ has also created a forgery that verifies on a new message. Thus, $\mathcal{B}$ breaks the unforgeability of SS. However, SS is a secure signature scheme, so $\mathcal{B}$ can break unforgeability with only negligible probability. We conclude that $\mathcal{A}$ can break revocation unforgeability with only negligible probability as well. □