Public-Key Encrypted Bloom Filters with Applications to Supply Chain Integrity

Florian Kerschbaum

SAP Research
Karlsruhe, Germany
florian.kerschbaum@sap.com

Abstract. Bloom filters provide a space- and time-efficient mean to check the inclusion of an element in a set. In some applications it is beneficial, if the set represented by the Bloom filter is only revealed to authorized parties. Particularly, operations data in supply chain management can be very sensitive and Bloom filters can be applied to supply chain integrity validation. Despite the protection of the represented set, Bloom filter operations, such as the verification of set inclusion, need to be still feasible. In this paper we present privacy-preserving, publicly verifiable Bloom filters which offer both: privacy for the represented set and public Bloom filter operations. We give security proofs in the standard model.

1 Introduction

Bloom filters provide a space- and time-efficient mean to check the inclusion of an element in a set in constant time. We apply them to supply chain integrity (see Section 2). Yet, they have many more applications in computer science, e.g. in databases [1,15] or networks [5].

We consider situations where the confidentiality of the set represented by the Bloom filter is important. Given an unprotected Bloom filter anyone can check for the inclusion of an element and maybe even enumerate all included elements. In many scenarios this is an undesired property, e.g. when the Bloom filter is stored or used by untrusted service provider [1,15]. The content (i.e. its bit mask representing the contained set) of the Bloom filter should remain private. This is particularly true in supply chain integrity where there are risks of industrial espionage [9,28].

Our idea starts by encrypting the Bloom filter content. Regular encryption renders the Bloom filter content useless. We therefore use a special, carefully crafted form of encryption: public-key and (partially) homomorphic. Now, only the private-key holder can access the Bloom filter content, but in order for the encrypted Bloom filter to be useful we need to still enable regular operations on it despite the encryption.

First, we enable the public-key holder to add elements to the Bloom filter by encrypting them – without interaction. Second, we enable the public-key holder to verify the inclusion or exclusion of an element – also without interaction.

Y. Li (Ed.): Data and Applications Security and Privacy XXV, LNCS 6818, pp. 60–75, 2011.

For these purposes we exploit the homomorphism of the encryption scheme to evaluate the Bloom filter operations and then employ zero-knowledge proofs (ZKP) [16] for validating the result. Our ZKPs guarantee that the private-key holder cannot make false claims about the Bloom filter content, yet the public-key holder will learn nothing beyond the validity of the claims. We emphasize that our secured Bloom filter operations can still be computed and verified in constant time.

We propose to apply these Bloom filters to supply chain integrity. Several important supply chain integrity checks can be reduced to set in- or exclusion. Any participant in the supply chain – whether supplier or customer – can verify using our privacy-preserving Bloom filters set inclusion and thereby e.g. product authenticity. Most importantly, no such check will violate any supplier's desire for privacy.

In summary, our contributions are

- the adaptation of a *public-key encryption* scheme for Bloom filters
- *non-interactive operations* for element addition, element inclusion or exclusion verification and filter content comparison
- *security proofs* in the standard model

The remainder of the paper is structured as follows. In the next section we present our example application of supply chain integrity and its security requirements. In Section 3 we present our building blocks of Bloom filters, public key encryption schemes and ZKPs. We describe our main result – a public-key encrypted Bloom filter – in Section 4. In Section 5 we review related work before we conclude the paper in Section 6.

2 Problem: Supply Chain Integrity

Supply chain integrity refers to the integrity of the flow of goods through a supply chain. This integrity can, e.g., be compromised by the introduction of counterfeit products or by the distribution of genuine products on gray markets. The sale of counterfeit products alone costs the United States an estimated 200 billion dollars annually [32].

Clearly, tracking of items and increased visibility of items throughout the supply chain help protecting supply chain integrity [30]. Nevertheless, this tracking also implies a number of novel security and privacy risks [28]. Given detailed information about one's supply chain operation one can infer strategic relationships, business volumes or planned promotions. Companies are therefore very reluctant to disclose this information despite its benefits [9].

In this paper we present secure methods for checking supply chain integrity that disclose nothing but the validity of the integrity check. We assume a generic model for item-level tracking in supply chains [30]. Each item is equipped with an unique identifier. Let $I = \{i_0, \ldots, i_n\}$ be the set of item identifiers. Also each supplier has an unique identifier. Let $S = \{s_0, \ldots, s_m\}$ be the set of supplier identifiers.

As an item i progresses through the supply chain it is handled by a number of different suppliers s. We can perform a number of simple, yet efficient checks on this process.

As a first application we can collect the set S_i of suppliers that have handled an item i. We create a Bloom filter d that represents the set of suppliers and transport it along with the item. This transport can be electronic in an accompanying network message (advanced shipping notification) or even on the item, e.g. an RFID tag [11]. Before a supplier ships the item to another supplier it adds the new supplier to the Bloom filter.

Given this Bloom filter d we can perform two distinct checks. First, we can compare the set S_i against a black list S_{bl} of known violators. These violators can be e.g. companies dealing on grey markets.

$$\forall s \in S_{bl}.s \notin S_i$$

Second, we can check the set S_i against a white list S_{wl} of mandatory suppliers. These suppliers can be e.g. the authentic manufacturers of the item.

$$\forall s \in S_{wl}.s \in S_i$$

For the technical implementation we need to be able to check whether an element s is in a set S_i. As already mentioned, Bloom filters offer a space- and time-efficient mean for this operation. We just need to protect the confidentiality of the represented set.

As a second application we can collect the set I_s of items that a supplier s has handled. We again create a Bloom filter d, but maintain it at one supplier. Each time this supplier handles an item i it adds it to the Bloom filter.

This time we can perform another check. Given two Bloom filters d_1 and d_2 at two suppliers s_1 and s_2, respectively, we can compare whether they encode the same set I. If they do, we are assured that there is no intermediate diversion of the flow of goods between the two suppliers.

$$I_{s_1} = I_{s_2}$$

For the technical implementation we need to be able to compare Bloom filter contents. This may seem simple, but we encrypt the Bloom filter contents using IND-CPA secure encryption [20], such that an equality comparison of the ciphertexts will fail.

Figure 1 exemplarily depicts these checks in a supply chain and how they capture illegitimate items. There are five suppliers s_1 to s_5 and three items i_1 to i_3. Each item takes a different path through the supply chain. Supplier s_1 is on the white list, while supplier s_4 is on the black list. The final customer (or any participant of the supply chain) can perform the following exemplar checks: First, for authentic item i_1 it can check whether it has been handled by supplier s_1: $s_1 \in S_1$. Second, for authentic item i_2 it check whether it has not been handled by supplier s_4: $s_4 \notin S_2$. Third, it can compare the set I_1 of supplier s_1 to the set I_5 of supplier s_5. While this check succeeds in our example, it would fail if supplier s_4 would have sold the item on the grey market (and thereby avoid the second check).

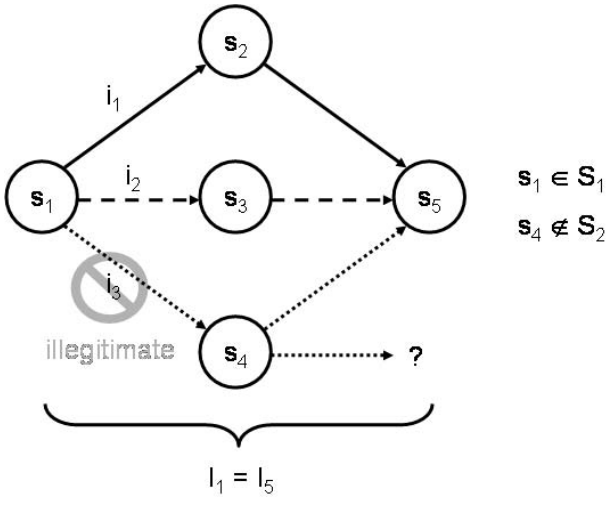

Fig. 1. Example Supply Chain with Illegitimate Item

2.1 Security Desiderata

Given a Bloom filter d we require a number of security properties. We distinguish only two parties: an authority and a supplier. The authority controls the Bloom filter. Its help is needed to perform the operations described above. The authority can be the manufacturer of an item or even an independent organization, such as an industry association. The supplier can add elements to the set and verify the checks described above, i.e. the supplier participates in the supply chain by handling goods and verifying the integrity of the supply chain. Loosely speaking, the goal of our algorithms is to protect against malicious suppliers. Most importantly, we do not distinguish between malicious and honest suppliers. This commonly made distinction is difficult to perform in practice, since the reliability of a supplier can vary over time and is difficult to assess. We assume that all suppliers may be malicious and may perform all operations on the Bloom filter.

Furthermore we assume that an attacker has full control over the network. We model the supply chain as a directed graph with vertices representing suppliers and edges representing transportation links. Items pass through the supply chain and along with each item i a Bloom filter for its set of suppliers S_i. Furthermore, each supplier s maintains a Bloom filter I_s of all of its items. An attacker may read and write any Bloom filter at any point in the graph. Given this powerful type of attacker some attacks cannot be prevented: disruption and cloning. We limit our protection goals to privacy and unlinkability.

Disruption. An attacker may simply destroy the Bloom filter and disrupt the communication. This cannot be prevented. Nevertheless, we can assume a default decision. Items without proper security checks can be considered illegitimate.

Then an attacker disrupting the supply chain cannot insert counterfeit items, but he can cause false positives resulting in a disruption of goods supply. Alternatively, items without proper security checks could be considered legitimate. This current practice prevents disruptions due to false positives, but the problem of counterfeits is prevalent.

Cloning. An attacker may simply copy the information of one Bloom filter to another. This attack is called cloning and is a common problem for anti-counterfeiting. There are no item-level (on-tag) countermeasures, but given a global data view, prevention is feasible [21,23,26,33]. We propose to augment both solutions, since our mechanism can protect against more supply chain integrity threats than just cloning.

Privacy. The content of a Bloom filter (i.e. the represented set) should remain private. Given any Bloom filter d an attacker should not be able to tell whether an element e is in the set or not (except with negligible probability). Even given several successful checks of inclusion or exclusion for elements e_i, an attacker should not be able to tell whether an element e' ($\forall i.e' \neq e_i$) is in the set or not (except with a small probability of false positives). Furthermore, given several successful checks of equality or inequality of sets, an attacker should still not be able to tell.

Unlinkability. An attacker should not be able to link a Bloom filter before and after the addition of an element. Given a pair of Bloom filters d_0 and d_1, an element e and a randomly chosen Bloom filter $d_b \in \{d_0 \cup \{e\}, d_1 \cup \{e\}\}$ with the element e added, an attacker should not be able to tell the random choice b (except with negligible advantage). This prevents an attacker from tracing items through the supply chain. It augments our privacy requirement in preventing supply chain espionage.

3 Background

3.1 Bloom Filter

Bloom filters [3] provide a space- and time-efficient mean to check the inclusion of an element in a set. An empty Bloom filter b consists of m bits, all set to 0, and k hash functions f_i ($0 \leq i < k$). We write b_j ($0 \leq j < m$) for the j-th bit of Bloom filter b. Bloom filters support the operations $add(x)$ for addition of element x to the set and $test(x)$ to test for inclusion of element x.
$Create(m)$: m bits ($0 \leq j < m$) are set to 0

$$\forall j.b_j = 0$$

and k hash functions f_i ($0 \leq i < k$) are published

$$\forall i.f_i : \{0,1\}^* \mapsto \{0,\ldots,m-1\}$$

$Add(x)$: The element x is hashed with all k hash functions f_i and the k bits at the resulting indices l_i are set to 1.

$$\forall i.l_i = f_i(x) \wedge b_{l_i} = 1$$

$Test(x)$: Again, the element x is hashed with all k hash functions f_i and if all k bits at the resulting indices l_i are set, then the test function returns true.

$$\bigwedge_{i=0}^{k-1} b_{f_i(x)}$$

Using Bloom filters false positive are possible, but false negatives are not. The more elements are added to the set, the more likely false positives are. Given the number n of elements to be added and a desired maximum false positive rate p, one can compute the necessary size m of the Bloom filter as [3]

$$m = -\frac{n \ln p}{\ln 2}$$

3.2 Goldwasser Micali Encryption

Goldwasser-Micali (GM) encryption [17] is a public-key, semantically-secure (IND-CPA), homomorphic encryption scheme. Its plaintext length is only 1 bit. GM encryption uses quadratic residuosity modulo a composite of two large primes p and q. A quadratic residue r is a number, such that there exists a number s: $s^2 = r \bmod n$. GM encodes a 1 as a quadratic non-residue and a 0 as a quadratic residue. Particularly, the quadratic non-residues are pseudo quadratic residues, i.e. their Jacobi symbols are all 1. Note that differentiating pseudo quadratic residues and quadratic residues implies factoring.

Let $n = pq$ be the composite of two large primes and v be pseudo quadratic residue. The public key is n, v and the private key is p and q. To encrypt a 0 one chooses a random number r and computes $r^2 \bmod n$ (a quadratic residue). To encrypt a 1 one also chooses a random number r and computes $vr^2 \bmod n$ (a quadratic non-residue). To decrypt one computes whether it is a quadratic residue.

We can summarize the operations as follows
KeyGen(κ): Let κ be a security parameter. Given κ generate the private key $sk = \{p, q\}$ and the public key $pk = \{n = pq, v\}$.
Encrypt(x, pk): Given plaintext x and public key pk produces ciphertext c.
Decrypt(c, sk): Given ciphertext c and private key sk produces plaintext x.

Let $E(x)$ denote encryption of x under GM public key pk. Multiplying two ciphertexts, e.g. $E(x) \cdot E(y)$, results in an encryption of the exclusive-or (XOR) denoted by $\oplus$.

$$E(x) \cdot E(y) = E(x \oplus y)$$

GM encryption is semantically-secure (IND-CPA) [20], i.e. one cannot infer from a ciphertext and the public key whether the ciphertext has a specific plaintext, e.g. by encrypting the plaintext and then comparing it.

3.3 Sander Young Yung Technique

Sander, Young and Yung operate on GM encryptions and allow the computation of one logical AND operation [29]. Recall that we can perform any number of logical XOR operations on the ciphertexts. A ciphertext $E(x)$ is expanded as follows.

Expand(c, pk): Given ciphertext $c = E(x)$ and public key pk compute σ_i. We repeat this operation u times ($0 \leq i < u$).

1. Flip a fresh random coin $r_i \in \{0,1\}$ ($i = 1, \ldots, u$).
2. Choose plaintext e_i according to the random coin and set

$$\sigma_i \leftarrow E(e_i) = \begin{cases} E(x) \cdot E(1) = E(x \oplus 1) & \text{if } r_i = 0 \\ E(0) & \text{if } r_i = 1 \end{cases}$$

The result is a u-length vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)$ which we call expanded ciphertext. If $x = 1$, then $x \oplus 1 = 0$ and $e_i = 0$. Then also $\sigma_i = E(0)$ for $i = 1, \ldots, u$. Otherwise, if $x = 0$, e_i is randomly distributed in $\{0,1\}$ and σ_i is a GM ciphertext of a random bit.

We can now compute a logical AND of two expanded ciphertexts $\boldsymbol{\sigma}$ (for $E(x)$) and $\boldsymbol{\rho}$ (for $E(y)$). We denote $\sigma_i = E(e_i)$ and $\rho_i = E(d_i)$. Logical AND is performed by pair-wise multiplication of the elements of the expanded ciphertext vectors: $\tau_i = \sigma_i \cdot \rho_i$. If $x \wedge y = 1$, then $\tau_i = E(c_i) = E(e_i) \cdot E(d_i) = E(e_i \oplus d_i) = E(0 \oplus 0) = E(0)$ for $i = 1, \ldots, u$, but if $x \wedge y = 0$, then c_i remains randomly distributed in $\{0,1\}$, since at least one of e_i or d_i is randomly distributed in $\{0,1\}$. Therefore $\boldsymbol{\tau}$ is the expanded ciphertext of $x \wedge y$. In order to decrypt an expanded ciphertext $\boldsymbol{\sigma}$ one decrypts each element $D(\sigma_i) = e_i$. If $e_i = 0$ for $i = 1, \ldots, u$, then the final plaintext $x = 1$; otherwise $x = 0$. There is a 2^{-u} probability that it is falsely decrypted as 1, since for an expanded ciphertext $\boldsymbol{\sigma}$ of $x = 0$ the plaintexts e_i are randomly distributed in $\{0,1\}^u$.

3.4 Quadratic Residuosity Zero-Knowledge Proofs

A simple proof that a ciphertext has plaintext 0 is to present a root s ($s^2 = r$). It can be verified by squaring s and is zero-knowledge, since it does not reveal the secret key p and q. Furthermore, if r is a quadratic non-residue, no such s exists.

Proof-QR(r):
Common input: r, $n = pq$
Prover's secret input: p, q

1. The prover outputs s.
2. The verifier accepts, if $s^2 = r$.

Nevertheless, this proof cannot be used to prove that a ciphertext has plaintext 1. If the prover claims that there is no root s, there is no way for the verifier to check it. In [10] Fiat and Shamir present a zero-knowledge proof (ZKP) that

r is a quadratic residue. The proof is analogous to the general ZKP for graph isomorphism by Goldreich, Micali and Widgerson in [16]. Furthermore in [16] they present a ZKP for graph non-isomorphism. We adapt this proof to quadratic residues and present a ZKP that r is a quadratic non-residue. We present its interactive form.

Proof-QNR(r):
Common input: r, $n = pq$
Prover's secret input: p, q

1. The verifier uniformly chooses a random number s and a bit $b \in \{0, 1\}$. If $b = 0$, then the verifier sends s^2 to the prover. If $b = 1$, then the verifier sends rs^2 to the prover.
2. The prover outputs a guess b' of b. The prover also sends a guess s' of s.
3. The verifier accepts if $b' = b$ and $s' = s$.

For a ZKP one has to prove three properties: (honest-verifier) zero-knowledge, completeness and soundness. Zero-knowledge means that the verifier learns nothing about the secret input of the prover. We can do so by showing a simulator of the verifier's view from its input (including random coin tosses) and output (of a successful proof). In this case, the simulator is particularly simple, since it simply mirrors the verifier's random choices b and s.

Completeness means that if r is indeed a quadratic non-residue an honest verifier will always accept. Clearly, if r is a quadratic non-residue then rs^2 is a quadratic non-residue, but s^2 is always a quadratic residue. Therefore the prover can distinguish the choice b by computing quadratic residuosity.

Soundness means that if r is not a quadratic non-residue, i.e. $t^2 = r$ an honest verifier will reject with high probability. If $b = 1$ and $t^2 = r$, then there exist a $s' = st$, such that $s'^2 = rs^2$. The message from the verifier is therefore indistinguishable to the prover for both cases of b. The probability of a right guess b' is then at most $\frac{1}{2}$.

In order to increase the probability for rejecting the ZKP in case of a quadratic residue we can repeat the above ZKP n times in parallel. The probability of a false accept is then 2^{-n}.

Furthermore, we can apply the technique by Blum, Feldman and Micali to make the ZKP non-interactive [4]. Given access to a common random string we can simulate the messages from the verifier. In our case it is critical to not simulate the random choices b, but just the messages themselves, i.e. the verifier sends a sequence of numbers u. We can non-interactively verify the correct guess of b by b' using s'. If the verifier sends a quadratic non-residue u (which he does with probability $\frac{1}{2}$) and r is a quadratic residue ($t^2 = r$), then there exists no s', since ur^{-1} is quadratic non-residue.

3.5 Shuffle Zero-Knowledge Proof

In addition to the quadratic residuosity ZKPs we need a further ZKP. Let $\boldsymbol{\sigma}$ be a u-length vector of GM ciphertexts $E(e_i)$. Let π be a random permutation for

$1, \ldots, u$ and $\boldsymbol{\rho}$ be a u-length vector of GM ciphertexts with plaintext 0. We can compute a shuffle $\boldsymbol{\tau} = \pi(\boldsymbol{\sigma}) \cdot \boldsymbol{\rho}$, such that given $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ (but not the secret key) nothing is revealed about π.

A shuffle ZKP proves that $\boldsymbol{\tau}$ is indeed a permutation of $\boldsymbol{\sigma}$, i.e. there exist π and $\boldsymbol{\rho}$.

Proof-Shuffle($\boldsymbol{\sigma}$, $\boldsymbol{\tau}$):

Common input: $\boldsymbol{\sigma}$, $\boldsymbol{\tau}$

Prover's secret input: π, $\boldsymbol{\rho} = (E(0), \ldots)$, such that $\boldsymbol{\tau} = \pi(\boldsymbol{\sigma}) \cdot \boldsymbol{\rho}$.

Groth and Ishai present a shuffle ZKP that has sub-linear communication complexity [18]. Sub-linear communication complexity means that less than u elements are transmitted.

4 Public-Key Encrypted Bloom Filter

In this section we present our main result: privacy-preserving, publicly verifiable Bloom filter. Due to our use of public-key encryption we call them public-key encrypted Bloom filter (PEBF). The basic idea of a PEBF is to encrypt each bit b_j of the Bloom filter using GM encryption. We present its operations $PEBF-Create(m, \kappa)$, $PEBF-Add(x)$, $PEBF-Test(x)$ and $PEBF-Compare(E(\boldsymbol{b}'))$.

PEBF-Create(m, κ):

1. Create a public-, private-key pair in the GM encryption scheme using $KeyGen(\kappa)$.
$$pk, sk \leftarrow KeyGen(\kappa)$$
2. Create a Bloom filter
$$b_j, f_i \leftarrow Create(m)$$
3. Encrypt each bit of the Bloom filter
$$E(b_j) \leftarrow Encrypt(b_j, pk)$$

Let $E(\boldsymbol{b})$ denote the element-wise encryption of $\boldsymbol{b} = (\ldots, b_j, \ldots)$ with the public key pk. The public part of the PEBF is $E(\boldsymbol{b}), f_i, pk$ and the private part is sk.

We give our first theorem that the public part of the PEBF does not leak any information about the content of the Bloom filter.

Theorem 1. *Let the adversary $\mathcal{A}$ choose two Bloom filter contents $\boldsymbol{b}_0 = (\ldots, b_{0,j}, \ldots)$ and $\boldsymbol{b}_1 = (\ldots, b_{1,j}, \ldots)$. Given a random choice β and the public part of a PEBF $E(\boldsymbol{b}_\beta), f_i, pk$, the probability that any adversary $\mathcal{A}$ outputs β is at most*

$$Pr\left[\mathcal{A}\left(\boldsymbol{b}_0, \boldsymbol{b}_1, E\left(\boldsymbol{b}_\beta\right), f_i, pk\right) = \beta\right] \leq \frac{1}{2} + \frac{1}{poly(\kappa)}$$

where $poly(\kappa)$ is an arbitrary polynomial in κ.

Proof. The proof is simple. Such an adversary $\mathcal{A}$ would contradict the IND-CPA security of GM encryption. We can simulate a successful adversary against GM encryption by embedding the challenge into the challenge of the adversary $\mathcal{A}$.

PEBF-Add(x):

1. Compute indices of Bloom filter for addition

$$l_i \leftarrow f_i(x)$$

2. Recompute each bit of the Bloom filter by replacing it with a plaintext 1 if it is set by the $Add(x)$ operation and re-randomizing if it is not set

$$E(b_j) = \begin{cases} E(1) & \text{if } \exists i.j = l_i \\ E(b_j) \cdot E(0) = E(b_j \oplus 0) & \text{otherwise} \end{cases}$$

We can rest assured that the public part of the PEBF before and after an addition does not leak any information about the added item. In fact, this is a corollary of Theorem 1.

Corollary 2. *Given two public parts $E(\boldsymbol{b}), f_i, pk$ for the same PEBF, but for Bloom filter contents $\boldsymbol{b}_0 = (\ldots, b_{0,j}, \ldots)$ and $\boldsymbol{b}_1 = (\ldots, b_{1,j}, \ldots)$, such that there exist an index h where $b_{0,h} \neq b_{1,h}$, the probability that any adversary $\mathcal{A}$ outputs h is at most*

$$Pr\left[\mathcal{A}\left(E\left(\boldsymbol{b}_0\right), E\left(\boldsymbol{b}_1\right), f_i, pk\right) = h\right] \leq \frac{1}{m} + \frac{1}{poly(\kappa)}$$

Proof. Construct an adversary $\mathcal{A}^\star$ for Theorem 1 by handing both ciphertexts $\boldsymbol{b}_0$ and $\boldsymbol{b}_1$ to adversary $\mathcal{A}$. If $\mathcal{A}$ guesses correctly, then $\mathcal{A}^\star$ guesses correctly.

PEBF-Test(x): Checking whether a PEBF contains an element x requires the private key sk. We construct a ZKP $PEBF-Test_{true}$ that x is contained within the public PEBF part $E(\boldsymbol{b})$, f_i, pk.

Common input: x, $E(\boldsymbol{b})$, f_i, pk
Prover's secret input: sk

1. Compute the set Bloom filter indices for x

$$l_i \leftarrow f_i(x)$$

2. Expand the ciphertext for each set Bloom filter bit

$$\boldsymbol{\sigma}_{l_i} \leftarrow Expand(E(b_{l_i}), pk)$$

3. Compute the logical AND of all set Bloom filter bits using the homomorphism

$$\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma}_{l_1} \cdot \ldots \cdot \boldsymbol{\sigma}_{l_k}$$

4. Proof in zero-knowledge that σ_j $(0 \leq j < u)$ is a quadratic residue

$$Proof - QR(\sigma_j)$$

Figure 2 depicts the process of ciphertext expansion on a PEBF.

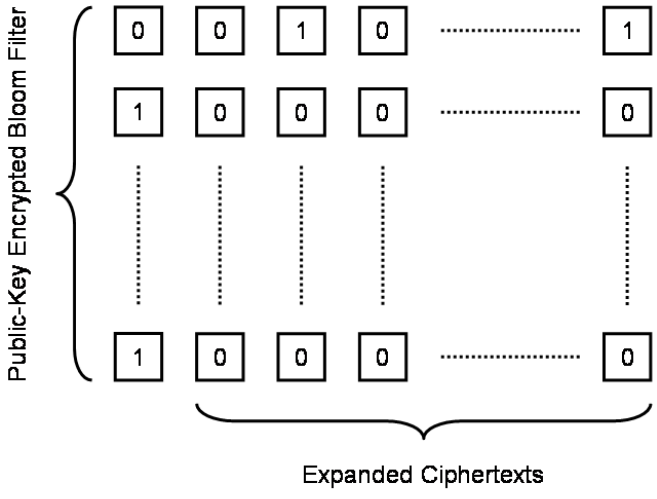

Fig. 2. Public-Key Encrypted Bloom Filter and Ciphertext Expansion

Theorem 3. *The zero-knowledge proof* $PEBF{-}Test_{true}$ *is honest-verifier zero-knowledge, complete and sound.*

Proof. For honest-verifier zero-knowledge we need to show a simulator for the view of the verifier. The simulator computes steps 1 to 3. It then invokes u times the simulator for *Proof-QR*(s^2).

We emphasize that the proof reveals that σ_i is a quadratic residue and (w.h.p.) that $E(b_{l_i})$ is a quadratic non-residue, but this is implied by the output of the ZKP.

For completeness we need to show that if $test(x) = true$, then $PEBF - Test_{true}$ is accepted by an honest verifier. If $test(x) = true$, then $E(b_{l_i})$ is a quadratic non-residue, $\sigma_{l_i,j}$ is a quadratic residue and consequently all σ_j are quadratic residues.

For soundness we need to show that if $test(x) = false$, then $PEBF{-}Test_{true}$ will be reject by an honest verifier with high probability. If $test(x) = false$, then there exist an index h $(0 \leq h < k)$, such that $l_h = f_h(x)$ and $b_{l_h} = 0$. Then $\sigma_{l_h,j}$ is (uniformly) randomly distributed in $\{E(0), E(1)\}$ and so is σ_j. Then at least one ZKP for quadratic residuosity will fail with probability $1 - 2^{-u}$.

In order to prove that an element x is not contained in a PEBF we need to prove that at least one index of $\boldsymbol{\sigma}$ has a quadratic non-residue. Unfortunately, knowing that σ_j is a quadratic non-residue may imply (w.h.p.) that (one specific) $b_j = 0$. Simply assume that the random choices in the *Expand*() operation, are such that the ciphertext of only one $E(b_j)$ is used and the others are fixed to $E(0)$.

We therefore need to construct a more complicated ZKP $PEBF - Test_{false}$.

1. Perform steps 1 to 3 as in $PEBF - Test_{true}$.
2. Choose a random permutation π of $(1, ldots, u)$ and a u-length vector of ciphertexts $\boldsymbol{\rho} = (E(0), \ldots)$. Compute
$$\boldsymbol{\tau} \leftarrow \pi(\boldsymbol{\sigma}) \cdot \boldsymbol{\rho}$$
3. Proof in zero-knowledge that $\boldsymbol{\tau}$ is a shuffle of $\boldsymbol{\sigma}$.
$$Proof - Shuffle(\boldsymbol{\sigma}, \boldsymbol{\tau})$$
4. Reveal an index h, such that τ_h is a quadratic non-residue and prove it in zero-knowledge
$$Proof - QNR(\tau_h)$$

Theorem 4. *The zero-knowledge proof $PEBF - Test_{false}$ is honest-verifier zero-knowledge, complete and sound.*

Proof. The proof for the properties of completeness and soundness are analogous to the proof for $PEBF - Test_{true}$.

For honest-verifier zero-knowledge we give the following simulator. Uniformly choose a random h. For $\boldsymbol{\tau}$ choose a random permutation of the ciphertexts $\boldsymbol{\sigma}$ except for τ_h choose one with plaintext 1 (a quadratic non-residue). Note that we might replace a quadratic residue at index h. Invoke the simulator for $Proof - Shuffle(\boldsymbol{\sigma}, \boldsymbol{\tau})$. If the simulator fails, because we did replace a quadratic residue, then rewind and choose a new h. The choice of h will fall on a quadratic non-residue with probability $\frac{1}{2}$. Therefore we succeed with high probability. Then invoke the simulator for $Proof - QNR(\tau_h)$.

PEBF-Compare($E(\boldsymbol{b}')$): Let $E(\boldsymbol{b}')$ be the encrypted Bloom filter content for the same hash functions f_i. Using the secret key sk we construct a ZKP $PEBF - Compare$ that $\boldsymbol{b}$ of the public part of a PEBF is equal.

Common input: $E(\boldsymbol{b}')$, $E(\boldsymbol{b})$, pk

Prover's secret input: sk

1. Compute the negated, logical XOR of the two encrypted Bloom filter contents using the homomorphism of the encryption scheme
$$E(\boldsymbol{b}'') \leftarrow E(\boldsymbol{b}) \cdot E(\boldsymbol{b}') \cdot E(1^m) = E(\boldsymbol{b} \oplus \boldsymbol{b}' \oplus 1^m)$$
2. Expand the ciphertext for each Bloom filter bit $(0 \leq i < m)$
$$\boldsymbol{\sigma}_i \leftarrow Expand(E(b_i''), pk)$$
3. Compute the logical AND of Bloom filter bits using the homomorphism
$$\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma}_0 \cdot \ldots \cdot \boldsymbol{\sigma}_{m-1}$$
4. Proof in zero-knowledge that σ_j $(0 \leq j < u)$ is a quadratic residue
$$Proof - QR(\sigma_j)$$

Theorem 5. *The zero-knowledge proof $PEBF - Compare$ is honest-verifier zero-knowledge, complete and sound.*

Proof. The proof for honest-verifier zero-knowledge is equal to the proof for honest-verifier zero-knowledge for $PEBF - Test_{true}$. We can use the same simulator.

For completeness we need to show that if $\boldsymbol{b} = \boldsymbol{b}'$, then $PEBF - Compare$ is accepted by an honest verifier. If $\boldsymbol{b} = \boldsymbol{b}'$, then $\boldsymbol{b}'' = 1^m$ and $\boldsymbol{\sigma}$ are all quadratic residues.

For soundness we need to show that if $\boldsymbol{b} \neq \boldsymbol{b}'$, then $PEBF - Compare$ will be reject by an honest verifier with high probability. If $\boldsymbol{b} \neq \boldsymbol{b}'$, then $\boldsymbol{b}''$ contains a 0 and $\boldsymbol{\sigma}$ contains a quadratic non-residue with probability $1 - 2^{-u}$. Consequently, at least one ZKP $Proof - QR(\sigma_j)$ will be rejected with high probability.

The construction of a ZKP that $\boldsymbol{b} \neq \boldsymbol{b}'$ follows the same ideas as ZKP $PEBF - Test_{false}$. We omit it for brevity.

5 Related Work

Our work is related to cryptographically secure Bloom filters [1,15,25], private set intersection [6,7,8,12,19,22] and anti-counterfeiting [2,21,23,24,26,27,31,33].

Cryptographically protected Bloom filters have been proposed before [1,15,25]. Nevertheless, the type of protection differs significantly from our approach.

In [1,15] Bloom filters are used for securely searching documents. It enables checking whether a document contains certain keywords without disclosing all of them. Their protection mechanism is to compute the hash function as a cryptographic pseudo-random function. This prevents reversing the Bloom filter, but it also prevents non-interactively adding an element which we enable.

In [25] an interactive protocol for securely checking set inclusion via Bloom filters without disclosing the Bloom filter content or the checked element. They also do not enable non-interactive (or even privacy-preserving) element addition. They use blind signatures in order to protect the Bloom filter content.

A related problem is private set intersection. Given two parties, each input a set of elements, privately compute the intersection of these two sets without disclosing either set. The first protocol secure in the semi-honest model has been presented in [22]. Efficiency improvements have been made in [12]. The malicious model has been first considered in [19] and further efficiency improvements have been made in [7,8]. An authority to certify the sets has been proposed in [6]. Note that – as opposed to all work on private set intersection – our operations work non-interactively. This also makes the distinction between semi-honest and malicious adversaries less applicable. Our security definitions are closer to public-key encryption.

The benefits of item tracking for anti-counterfeiting have been first recognized in [31]. They already outline the two basic approaches beyond item identification itself: on-tag and in-network.

In-network protection collects information about all items and correlates it. It can prevent cloning attacks. A statistical method based on detection of low probability events is presented in [23]. This method requires sharing of information. A similar method that protects this information using secure multi-party computation has been presented in [33]. A deterministic method for detecting integrity violations has been presented in [26]. It also requires sharing of information. A secure variant using cryptographic hashing has been presented in [21].

On-tag protection only stores information on the RFID tag. Methods using more powerful RFID tags that support cryptographic hashing have been proposed first [24,27]. Recently, a method using only storage on the RFID tag has been described [2]. Our public-key encrypted Bloom filters (augmented with standard signatures) implement not only their full functionality, but surpass it in several aspects. First, we enable more checks than just path verification, such as our compare operation. Second, we provide security against the verifier of integrity considering an attacker that is part of the supply chain.

6 Conclusions

In this paper we have presented public-key encrypted Bloom filters. The content of the Bloom filter is encrypted using public-key, homomorphic encryption. Only the private-key holder can access the Bloom filter content. We enable the public-key holder to non-interactively add elements by encrypting them. Furthermore, we present zero-knowledge proofs for non-interactively verifying the inclusion or exclusion of an element and the equality of two Bloom filter contents.

Given such protected Bloom filters one can perform several privacy-preserving supply chain integrity checks. One can check the path of item through a supply chain against black lists, white lists or for equality. The public-key encryption protects the confidentiality of the Bloom filter content during all these operations.

There are a few possible improvements for future work. First, the set inclusion or exclusion zero-knowledge proofs reveal the element checked. This could be prevented by also encrypting it, but the homomorphism of existing (efficient) public-key encryption schemes is insufficient. When fully homomorphic encryption [14] becomes practical, it may provide a further avenue.

Second, the set inclusion or exclusion zero-knowledge proofs also require the knowledge of the ciphertext. Ideally the private-key holder could issue a security token without knowing the ciphertext in question. This could be done using searchable encryption, but the existing searchable encryption schemes do not support homomorphisms. Given improved, searchable encryption schemes, a new construction might become feasible.

Third, the bit-wise encryption of Goldwasser-Micali encryption is quite storage-intensive. While RFID tags with sufficient storage capacity – up to 64 KByte – exist [13], a reduction of the storage requirements would enable using cheaper RFID tags. Of course, this is no restriction for the collection of all handled items at one supplier.

References

1. Bellovin, S., Cheswick, W.: Privacy-Enhanced Searches Using Encrypted Bloom Filters. Cryptology ePrint Archive Report 2004/022 (2004)
2. Blass, E., Elkhiyaoui, K., Molva, R.: Tracker: Security and Privacy for RFID-based Supply Chains. In: Proceedinsg of the 18th Network and Distributed System Security Symposium, pp. 455–472 (2011)
3. Bloom, B.: Space/Time Trade-offs in Hash Coding with Allowable Errors. Communication of the ACM 13(7), 422–426 (1970)
4. Blum, M., Feldman, P., Micali, S.: Non-Interactive Zero-Knowledge and Its Applications. In: Proceedings of the 20th ACM Symposium on Theory of Computing, pp. 103–112 (1988)
5. Broder, A., Mitzenmacher, M.: Network Applications of Bloom Filters: A Survey. Internet Mathematics 1(4), 485–509 (2003)
6. Camenisch, J., Zaverucha, G.: Private Intersection of Certified Sets. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 108–127. Springer, Heidelberg (2009)
7. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient Robust Private Set Intersection. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 125–142. Springer, Heidelberg (2009)
8. De Cristofaro, E., Kim, J., Tsudik, G.: Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 213–231. Springer, Heidelberg (2010)
9. Eurich, M., Oertel, N., Boutellier, R.: The Impact of Perceived Privacy Risks on Organizations' Willingness to Share Item-Level Event Data Across the Supply Chain. Electronic Commerce Research 10(3-4), 423–440 (2010)
10. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
11. Finkenzeller, K.: RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification. John Wiley & Sons, Inc., Chichester (2003)
12. Freedman, M., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
13. Fujitsu. Fujitsu Develops World's First 64KByte High-Capacity FRAM RFID Tag for Aviation Applications. Press Release (2008), `http://www.fujitsu.com/global/news/pr/archives/month/2008/20080109-01.html`
14. Gentry, C.: Fully Homomorphic Encryption using Ideal Lattices. In: Proceedings of the 41st ACM Symposium on Theory of Computing, pp. 169–178 (2009)
15. Goh, E.: Secure Indexes. Cryptology ePrint Archive Report 2003/216 (2003)
16. Goldreich, O., Micali, S., Wigderson, A.: Proofs that Yield Nothing but Their Validity or All Languages in NP have Zero-Knowledge Proof Systems. Journal of the ACM 38(3), 690–728 (1991)
17. Goldwasser, S., Micali, S.: Probabilistic Encryption. Journal of Computer and Systems Science 28(2), 270–299 (1984)
18. Groth, J., Ishai, Y.: Sub-Linear Zero-Knowledge Argument for Correctness of a Shuffle. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008)
19. Hazay, C., Lindell, Y.: Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)

20. Katz, J., Lindell, Y.: Introduction to Modern Cryptography: Principles and Protocols. Chapman & Hall/CRC (2007)
21. Kerschbaum, F., Oertel, N.: Privacy-Preserving Pattern Matching for Anomaly Detection in RFID Anti-Counterfeiting. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 124–137. Springer, Heidelberg (2010)
22. Kissner, L., Song, D.: Privacy-Preserving Set Operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
23. Lehtonen, M., Michahelles, F., Fleisch, E.: How to Detect Cloned Tags in a Reliable Way from Incomplete RFID Traces. In: Proceedings of the IEEE RFID Conference, pp. 257–264 (2009)
24. Li, Y., Ding, X.: Protecting RFID Communications in Supply Chains. In: Proceedings of the ACM Symposium on Information, Computer and Communications Security, pp. 234–241 (2007)
25. Nojima, R., Kadobayashi, Y.: Cryptographically Secure Bloom-Filters. Transactions on Data Privacy 2, 131–139 (2009)
26. Oertel, N.: Tracking based product authentication: Catching intruders in the supply chain. In: Proceedings of the 17th European Conference on Information Systems (2008)
27. Ouafi, K., Vaudenay, S.: Pathchecker: an RFID Application for Tracing Products in Suply-Chains. In: Proceedings of the 5th Workshop on RFID Security (2009)
28. Santos, B., Smith, L.: RFID in the supply chain: panacea or pandora's box? Communications of the ACM 51(10), 127–131 (2008)
29. Sander, T., Young, A., Yung, M.: Non-Interactive CryptoComputing For NC1. In: Proceedings of the 40th Symposium on Foundations of Computer Science, pp. 554–567 (1999)
30. Sarma, S., Brock, D., Engels, D.: Radio frequency identification and the electronic product code. IEEE Micro 21(6), 50–54 (2001)
31. Staake, T., Thiesse, F., Fleisch, E.: Extending the EPC Network – The Potential of RFID in Anti-Counterfeiting. In: Proceedings of the 20th ACM Symposium on Applied Computing, pp. 1607–1612 (2005)
32. Waldbaum, M., Nguyen, X.: Using Creativity to Fight a $60 Billion Consumer Problem – Counterfeit Goods. Loyola Chicago Consumer Law Review 10(1), 88 (1998)
33. Zanetti, D., Fellmann, L., Capkun, S.: Privacy-preserving Clone Detection for RFID-enabled Supply Chains. In: Proceedings of the IEEE International Conference on RFID, pp. 37–44 (2010)