Single-bit Re-encryption with Applications to Distributed Proof Systems

Nikita Borisov
nikita@uiuc.edu
Electrical and Computer Engineering Dept.

Kazuhiro Minami
minami@uiuc.edu
Computer Science Dept.

University of Illinois at Urbana–Champaign
Urbana, IL 61801

ABSTRACT

We examine the implementation of the distributed proof system designed by Minami and Kotz [17]. We find that, although a high-level analysis shows that it preserves confidentiality, the implementation of the cryptographic primitives contains a covert channel that can leak information. Moreover, this channel is present with any traditional choice of public key encryption functions.

To remedy this problem, we use the Goldwasser-Micali cryptosystem to implement single-bit re-encryption and show how to make it free of covert channels. We then extend the primitive to support commutative encryption as well. Using this primitive, we design a variant of the Minami-Kotz algorithm that not only is free of covert channels, but also has additional proving power over the original design.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption; C.2.4 [**Computer Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Security

Keywords

Distributed proof systems, covert channels, re-encryption, commutative encryption, Goldwasser–Micali

1. INTRODUCTION

Recent years have seen the development of several distributed proof systems. These systems are generally used as a means of distributed authorization, where different agents supply different rules and statements that lead to an authorization decision being performed. These systems include Binder [6], the Grey system [1, 2], PeerAccess [24], and SD3 [13].

We look in particular at a proof system designed by Minami and Kotz [17, 18]. Their system had two innovative features: explicit policies for integrity and confidentiality of facts and rules, and a cryptographic protocol that allows mutually untrusting agents to nevertheless compose their proofs. In essence, the protocol consists of a party sending an encrypted fact to a principal who is not authorized to see it, who then performs proof operations with the encrypted fact, and then passes it to some other principal who *is* authorized to see it and who decrypts it.

A high level analysis of the protocol shows that it is secure, in that it does not reveal information about facts protected by confidentiality policy either directly or through inference. However, we analyze the implementation of the protocol and find that the cryptographic primitives used include a covert channel that can be used to compromise confidentiality. Furthermore, we note that *any* traditional asymmetric encryption primitive will have the same problem.

To address this issue, we use the Goldwasser–Micali encryption scheme [9] together with re-encryption, to build a primitive we call *single-bit re-encryption*. This primitive allows the asymmetric encryption of a single bit to be transformed into another, completely unlinkable with the original but preserving the bit value. This scheme is free of covert channels despite malicious actions of any of the parties. In addition to the Minami–Kotz algorithm, such a primitive may have other applications in the fields of electronic voting or online games.

We show how single-bit re-encryption can solve the covert channel problem in the Minami–Kotz algorithm. We need to slightly modify the MK algorithm because we cannot use single-bit re-encryption to construct nested encrypted values, as are used in MK. We instead design a commutative encryption scheme compatible with single-bit re-encryption, and show that the semantics of commutative encryption preserve the safety properties of the original MK algorithm. As a side benefit, commutative encryption allows for a greater scale of collaboration between mutually untrusting parties, increasing the proving power over the previous design.

The rest of the paper is organized as follows. In Section 2, we present an overview of the MK algorithm and show that it preserves each peer's confidentiality requirements using a high-level analysis. We examine the implementation details and demonstrate an inference attack using a covert channel in Section 3. In Section 4, we develop a single-bit re-encryption primitive, and use it to create a modified version of the MK algorithm in Section 5. Section 6 discusses other applications and related work and Section 7 concludes.

WPES'07, October 29, 2007, Alexandria, Virginia, USA.

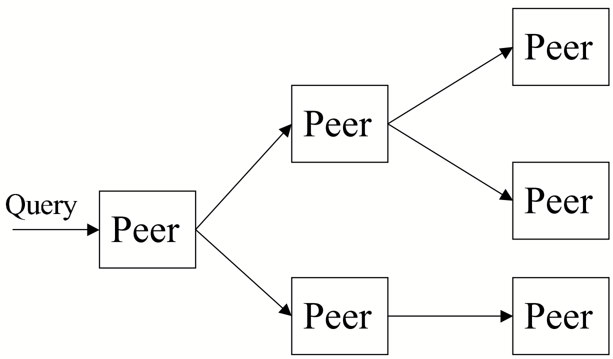


Figure 1: Distributed proving. Each arrow represents a query to a remote peer.

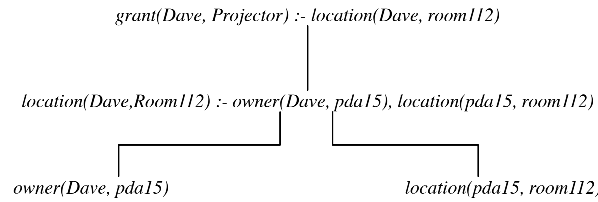


Figure 2: Sample proof tree.

2. BACKGROUND

In this section, we review the Minami-Kotz distributed proof construction algorithm (MK algorithm). Rather than review the full details of the algorithm, we focus on some key features and explain them with the help of examples; readers interested in a full description of the MK algorithm are referred to [17] for more details.

2.1 Structure of a peer server

A set of peers, each of whom consists of a knowledge base and an inference engine, perform the MK algorithm to build a proof in a decentralized way. Without loss of generality, we assume that peers are administered by different principals. The knowledge base of a peer stores both rules and facts. The inference engine provides an interface for handling queries from remote peers, and for issuing subqueries to remote peers; that is, receiving a query from another peer, the inference engine attempts to derive logical proofs justifying these queries using the facts in its local knowledge base and potentially interactions with remote parties as shown in Figure 1; this process is iterated at each peer handling a query. If the inference engine cannot construct a proof with remote peers, it returns a proof that contains a false value.

Rules and facts in a knowledge base are represented as a set of Horn clauses in Prolog. For example, an administrator of a meeting room defines an authorization policy that requires a requester P accessing a projector to be physically located at the meeting room (e.g., room112) as follows:

$$grant(P, projector) \text{ :- } location(P, room112)$$

The atom $location(P, room112)$ on the right side of the clause is the condition that must be satisfied to derive the granting decision $grant(P, projector)$ on the left. The administrator could define another rule to derive a requester's location from the location of a device owned by the requester as follows:

$$location(P, L) \text{ :- } owner(P, D), location(D, L)$$

If a user *Dave* issues a request to access a projector at *room112*, the proof tree in Figure 2 could be constructed based on the above rules. The intermediate nodes in the tree represent the rules and the two leaf nodes represent the facts. Notice that the variables P and D in those rules are replaced with constants *Dave* and *pda15* respectively.

2.2 Proof Decomposition

Multiple peers in different administrative domains can cooperate to handle a query in a peer-to-peer manner. This peer-to-peer interaction is guided by each peer's *integrity policies*, which specify sets of principals trusted to handle particular types of queries. For example, if Alice specifies the integrity policy $trust(location(P, L)) = \{Bob\}$, then she trusts Bob to accurately answer queries regarding the location of other entities. In the most basic case, the principal who issues a query trusts the principal who handles this query in terms of the integrity of the query result. As such, the handler principal need not disclose the entire proof tree that she generates, she needs only to return a proof that states whether the fact in the query was true. In general, however, the querier may not *completely* trust the query handler and thus her integrity policies might place constraints on the rules used by the handler to generate the proof tree. In this case, a more complete proof tree, whose nodes are digitally signed, would need to be returned by the handler. This way, the querier can verify that her integrity policies were respected.

Figure 3 describes one possible collaboration between a querier and handler. Suppose that host A, run by principal Alice, receives a query $?grant(Dave, projector)$ that asks whether Dave should be granted access to a projector owned by Alice. Since Alice's authorization policy in her knowledge base refers to a requester's location ($location(P, room112)$), Alice issues a query $?location(Dave, room112)$ to host B run by Bob. Alice chooses Bob because Bob satisfies Alice's integrity policy for queries of the type $location(P, L)$. Bob processes the query from Alice, because Alice satisfies Bob's confidentiality policy for location queries $location(P, L)$ as defined in Bob's policy $acl(location(P, L)) = \{Alice\}$. Bob derives the fact that Dave is in *room112* from the location of his device using the facts $location(pda15, room112)$ and $owner(Bob, pda15)$. However, he needs to return only a proof containing a single root node with the statement $location(Dave, room112)$, because Alice believes Bob's statement about people's location (i.e., $location(P, L)$) according to her integrity policy. The proof of the query, shown in Figure 2, is thus decomposed into two subproofs maintained by Alice and Bob.

2.3 Enforcement of Confidentiality Policies

Each fact provider maintains a set of *confidentiality policies* that determine which entities are authorized to receive the facts that she provides. These policies are enforced by encrypting a query result using the public key of an authorized receiver. Each query is accompanied by a list of upstream principals who could possibly receive the answer of the query; this enables the handler to choose an authorized

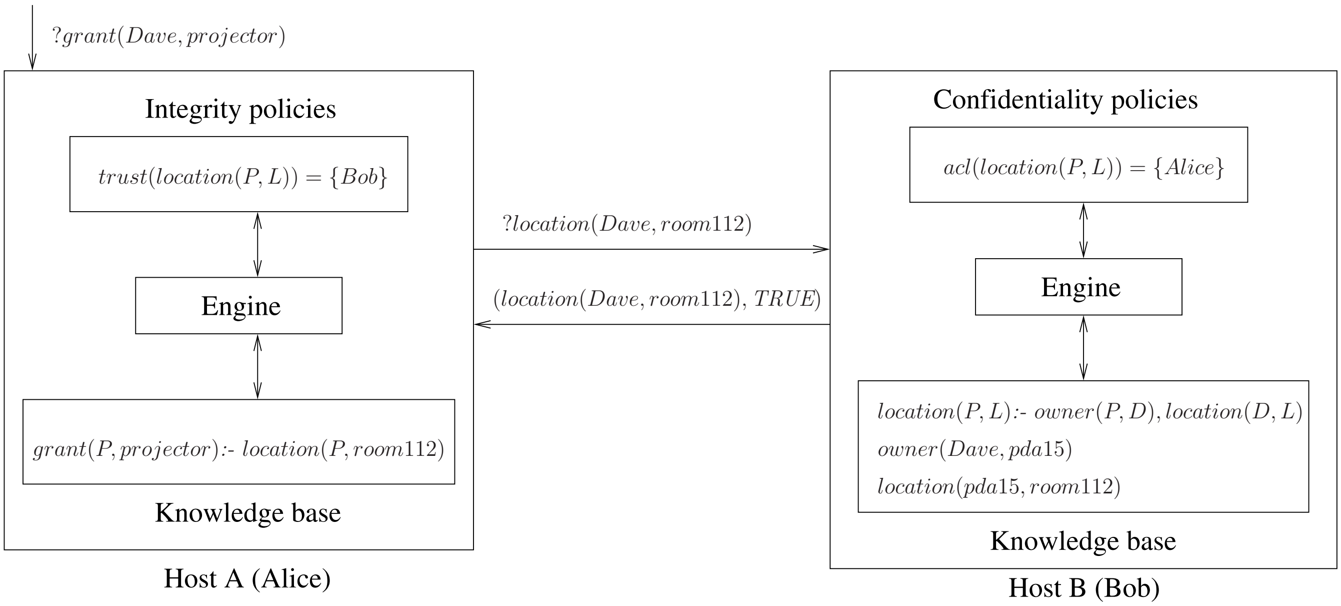


Figure 3: Remote query between two principals. Alice is a principal who maintains a projector, and Bob is a principal who runs a location server.

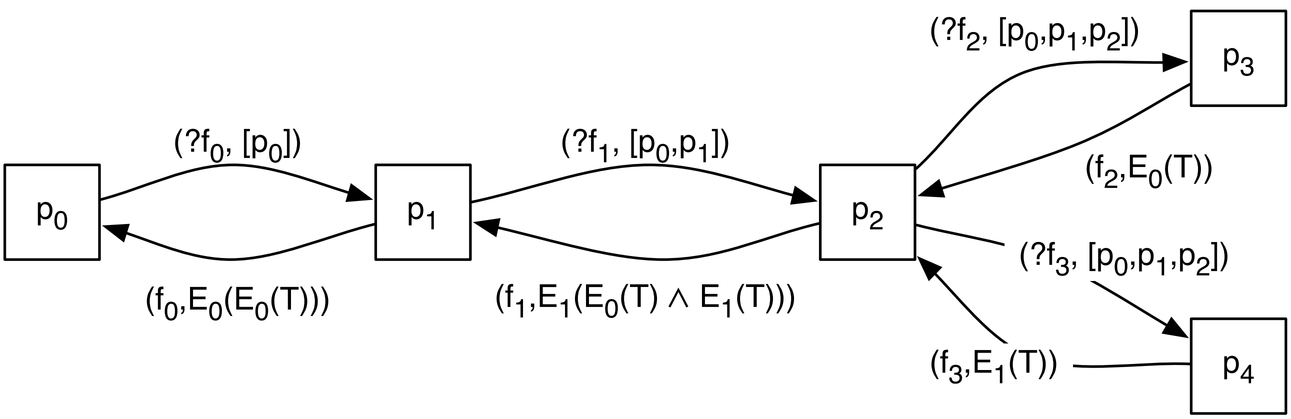


Figure 4: Enforcement of confidentiality policies. Each query is associated with a set of upstream principals that could receive a reply for the query. The first item in a proof tuple is a queried fact, and the second item is the validity of that fact, encrypted with the receiver's public key.

recipient from the list of upstream principals that satisfies her confidentiality policies. It is therefore possible to obtain an answer for some initial query even when some number of intermediate principals in the distributed proof do not satisfy the confidentiality policies of a fact provider. Figure 4 shows an example collaboration among principals p_0, p_1, p_2, and p_3. When principal p_0 issues an authorization query f_0 to principal p_1, p_1 issues a subsequent query f_1, which causes principal p_2's queries f_2 and f_3. Since the receiver principal of a proof might not be the principal who issues a query, a reply for a query is a tuple $(f, E_i(T))$ where f is a fact and $E_i(T)$ is an encrypted value with the receiver p_i's public key. (If the receiver principal cannot construct a proof for the query, the encrypted value contains a false value F instead.) The identity of a receiver principal, which is omitted for brevity in Figure 4, is associated with an encrypted value so that a principal who receives an encrypted fact can decide whether to attempt to decrypt that encrypted fact. We assume that, in this example, each principal who issues a query trusts the recipient of the query with the integrity of the fact being queried. For example, p_0's integrity policies contain a policy $trust(q_0) = \{p_1\}$.

Suppose that query f_1's result (i.e., true or false) depends on the results of queries f_2 and f_3, which are handled by principals p_3 and p_4, respectively, and that p_3 and p_4 choose principals p_0 and p_1, respectively, as receivers since p_2 does not satisfy their confidentiality policies. Principal p_2 cannot decrypt the results from principals p_3 and p_4, but p_2 knows that the query result of f_1 is the conjunction of the values in those encrypted results. Therefore, p_2 encrypts those results with the public key of principal p_1, which p_2 chose as a receiver, recursively.

Principal p_1 decrypts the encrypted result from p_2 and obtains the encrypted results originally sent from principals p_3 and p_4. Since p_1 is a receiver of the proof from p_4, p_1 decrypts the encrypted fact $E_1(T)$ that contains a true value. Since a query result for f_0 depends on the encrypted fact from p_3, principal p_1 encrypts it with p_0's public key according to p_1's confidentiality policy (i.e., $acl_1(f_0) = \{p_0\}$) and returns it to p_0. (If the proof from p_4 contains a false value, p_1 knows the query result of q_0 without decrypting p_3's proof. Thus, p_1 discards p_3's proof and returns a proof tree whose root node contains a false value.) The principal p_0 finally decrypts it and obtains an answer for query f_0. The key observation here is that principal p_0 is not aware of the fact that the query result is originally produced by principal p_3.

Notice that, in the MK algorithm, each principal must perform a decryption operation on the outermost layer of encryption on that fact. Therefore, if principal p_2 in Figure 4 chooses p_0 as a receiver, principal p_1 would not be able to decrypt $E_0(E_0(T) \wedge E_1(T))$, and then p_0 would be left with $E_1(T)$, which he could not decrypt. Therefore, it is impossible for p_0 to obtain an answer to the query f_0.

2.4 Safety property of the MK algorithm

The MK algorithm ensures the following safety property for confidential facts maintained by principals participating in its protocol.

PROPOSITION 1. *If a principal p_i publishes an encrypted fact for f, the principal that decrypted and read that fact must belong to p_i's confidentiality policy $acl_i(f)$.*

DEFINITION 1. *We say that an encrypted fact published by a principal has a logical dependency on an encrypted fact received by that principal if the principal uses the received fact to derive the published fact by applying some local rules.*

LEMMA 1. *Without performing global traffic analysis, a set of principals cannot detect the logical dependencies among encrypted facts published and received by a principal that does not belong to that set.*

PROOF. Suppose that two colluding principals p_0 and p_2 observe that p_1 receives fact $(f_1, E_0(T))$ and publishes fact $(f_0, E_0(T))$, as shown in Figure 5(a) and that p_1 derives the published fact $(f_0, E_0(T))$ by applying rule $f_0 \leftarrow f_1$. However, p_0 and p_2 cannot decide the logical dependency between those two encrypted facts since there might exist an extra principal p_3, which provides p_1 with another encrypted fact $(f_2, E_0(T))$, as shown in Figure 5(b). Principal p_1 could derive the published fact $(f_0, E_0(T))$ by applying another rule $f_0 \leftarrow f_2$. That is, principals p_0 and p_2 cannot eliminate the possibility of having such an extra principal without performing a global traffic analysis. In general, it is always possible to construct any nested form of an encrypted fact by introducing extra principals. □

THEOREM 1 (SAFETY OF THE MK ALGORITHM). *If a principal p_i maintains a confidential fact f in its knowledge base, a set of principals S that do not belong to $acl_i(f)$ cannot decide whether p_i has a fact f or not in its local knowledge base without performing a global traffic analysis.*

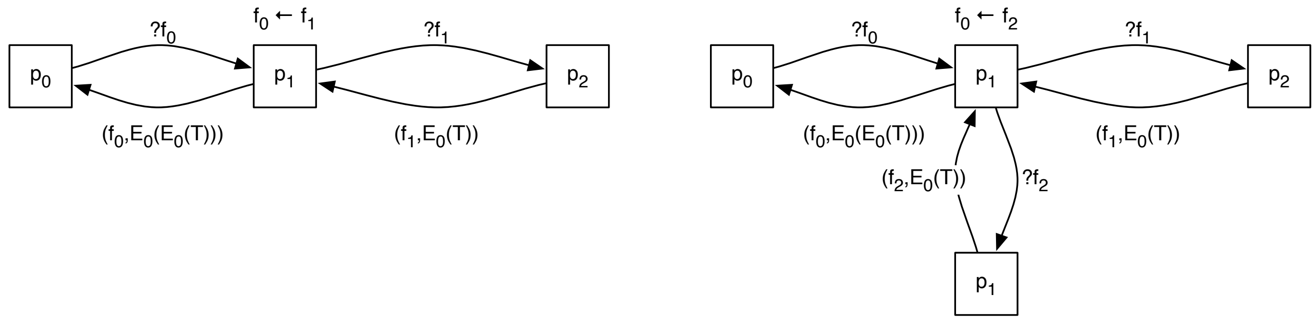
PROOF. The theorem follows directly from Proposition 1 and Lemma 1. □

3. ATTACK ON THE MK ALGORITHM

The proof of Lemma 1 assumes that the adversary cannot distinguish between cases (a) and (b) in Figure 5, as in both cases, p_0 receives $E_0(E_0(T))$ from p_1. However, the implementation of the underlying encryption primitives will violate this assumption. Minami and Kotz use RSA-OAEP [3] to encrypt the boolean values T and F. However, RSA-OAEP is a *randomized* encryption protocol; in essence, the encryptor chooses some random padding to add to the message before encryption. In this case, p_2 can choose the padding to be of a special format, previously agreed upon with p_0, represented by $\overline{E_0(T)}$. In this case, in situation (a), p_0 will receive $E_0(\overline{E_0(T)})$, whereas in situation (b), it receives $E_0(E_0(T))$, thus immediately distinguishing between the two situations.

This can lead to a breach of confidentiality. For example, consider the example in Figure 4. Suppose that p_2, p_3, and p_0 conspire to learn the value of f_3 in p_4's knowledge base.[1] Looking at a high-level execution of the protocol, they cannot determine this, because p_2 cannot decrypt $E_1(T)$ that it receives from p_3, and p_0 cannot tell if the result $E_0(E_0(T))$ was caused by the query to p_2 or some other query to another principal. However, if p_3 marks its response to p_2 as $\overline{E_0(T)}$, then p_2 will forward $E_1(\overline{E_0(T)} \wedge E_1(T))$ to p_1, who will forward $E_0(\overline{E_0(T)})$ to p_0. Any other value $E_0(T)$ received by p_1 from another principal would be distinguishable from the marked $\overline{E_0(T)}$, and thus the collection of principals is able to violate p_4's confidentiality policy.

[1]We assume here that $acl_4(f_3) = \{p_1\}$.



(a) Local view from an adversary

(b) Global view with an extra principal

Figure 5: Unlinkability among encrypted facts received and published by a principal. Principals p_0 and p_2 observe that p_1 receives encrypted fact $(f_1, E_0(T))$ and publishes $(f_0, E_0(T))$. However, p_0 and p_2 cannot decide the dependency between those encrypted facts, as shown in Figure 5(a), since there might exist an unknown extra principal p_3, which provides p_1 with another encrypted fact $(f_2, E_0(T))$, as shown in Figure 5(b).

Switching to an encryption scheme that is deterministic would eliminate this problem, but introduce another. A deterministic public-key encryption scheme is subject to a dictionary attack: an adversary in possession of $E_i(X)$ can compute $E_i(x)$ for $x \in D$, some dictionary, until he finds a match where $E_i(X) = E_i(x)$. This is particularly problematic in this case, since most encryptions are of two values—True and False. Randomized encryption provides semantic security [10] of encrypted values, but introduces a covert channel. We next develop a primitive to help eliminate this channel.

4. SINGLE-BIT RE-ENCRYPTION

The crux of the attack is that encrypted values are recognizable by colluding parties as they are passed between principals in a distributed proof. To address this issue, we actually need to solve two problems. First, we need to make sure that encrypted values themselves carry no identifying markings; in other words, when $E_i(T)$ is forwarded from p_j to p_l, it should be impossible to tell if this is the same value as was sent by p_k to p_j. Second, we need to make sure that after decryption, this is still true. During normal operation, a principal will only encrypt values of the form $E_i(T)$ or $E_i(F)$. Therefore, encryption of other values, e.g. $E_i(\text{"}signal\text{"})$, would violate the confidentiality constraints of the algorithm. Therefore, we must ensure that encrypted values are restricted to a small domain.

4.1 Goldwasser–Micali

We start by addressing the second problem. We use the encryption scheme designed by Goldwasser and Micali [9]. They designed a semantically secure public key encryption scheme based on the quadratic residuosity problem that supports encryption of a single bit. This restriction makes it impractical for most uses, as it expands the ciphertext by a large factor; however, the single bit restriction is exactly what we need. We review the scheme below.

The scheme takes place in the ring of integers modulo $n = pq$, where p and q are two primes. As in RSA, n is a public parameter, but p and q are kept private. A number a *quadratic residue* (QR) modulo n if there exists a b such that $a \equiv b^2 \pmod n$. It is easy to see that a is a QR mod n if and only if a is a QR both mod p and mod q, otherwise it's considered a *non-quadratic residue* (NQR).

The Jacobi symbol of a number a mod a prime p, written as (a/p), is $+1$ if a is a QR mod p and -1 otherwise. The Jacobi symbol of a mod $n = pq$, $(a/n) = (a/p)(a/q)$. (a/n) is $+1$ whenever a is a QR mod n, and whenever a is an NQR mod p and mod q; otherwise it is -1. The Jacobi symbol can be efficiently computed using a modified GCD algorithm, even if the factorization of n is not known [7]. However, the *quadratic residuosity assumption* states that if $(a/n) = +1$, it is intractable to tell whether a is a QR or an NQR without knowing the factorization of n. If the factorization of n is known, it is easy to check if a is a QR by computing:

$$a^{\frac{p-1}{2}} \pmod p \text{ and } a^{\frac{q-1}{2}} \pmod q$$

These numbers will be equal to the Jacobi symbols (a/p) and (a/q), respectively, and hence a will be a QR if both of those numbers are 1.

The encryption scheme proceeds as follows. The public key consists of n and a number x that is an NQR modulo n with Jacobi symbol $+1$. The private key consists of p and q. To encrypt a bit b with the public key (n, x), the sender picks a random y and sends $y^2x^b \pmod n$. The result will be a QR if $b = 0$ and an NQR if $b = 1$. The owner of the private key can use p and q to check which is which by the above method, whereas anyone else cannot tell, assuming the quadratic residuosity assumption holds.

4.2 Re-encryption

While Goldwasser–Micali restricts the domain to a single bit, the choice of y still presents a covert channel and the encrypted message can be recognized by a colluding receiver, even if not in possession of the private key. We can defend against this by having each sender *re-encrypt* the value $E_i(X)$. Re-encryption is the technique of producing an equivalent, but unlinkable, encryption of a ciphertext without knowing the private key. ElGamal re-encryption is the most common, frequently used to generate secret shuffles in mix networks [8, 11, 12, 19]. However, Goldwasser–Micali also supports re-encryption: given an encrypted value a, a sender can pick y' at random and send $a(y')^2 \pmod n$. The value will be a QR if and only if a is a QR, so it is an equivalent encryption of the same bit. To show that it is a secure re-encryption, it remains to show that it is unlinkable to the original value.

Suppose an adversary picks two QRs, a_1 and a_2 and is given b, a re-encryption of one of the two. As long as $\gcd(a_i, n) = 1$, there will exist y_1, y_2 such that $a_1 y_1^2 \equiv a_2 y_2^2 \equiv b \pmod{n}$. Therefore, the adversary cannot tell whether b is a re-encryption of a_1 or a_2. By a similar argument, if a_1 and a_2 are both NQRs (with Jacobi symbol $+1$), the adversary cannot distinguish the re-encryption of the two. Therefore, given a re-encryption of a ciphertext, an adversary can (at best) tell whether the ciphertext encrypts a 1 or a 0, but no other information, thus eliminating covert channels.

Prior to re-encryption, it is necessary to perform additional checks to ensure that the adversary is not deviating from the Goldwasser–Micali scheme in order to create a covert channel. First, we need to check that $\gcd(a, n) = 1$. (Of course, this test fails only if $a = 0$ or if a is a multiple of p or q, hence leading to an easy factorization of n.) Second, we need to verify that the Jacobi symbol $(a/n) = +1$, since otherwise a would retain a negative Jacobi symbol after re-encryption; fortunately, this can be done efficiently as mentioned above. A final attack to defend against is an adversary who picks n to be not an RSA modulus. For example, if n is a product of four primes, $pqrs$, then it is possible to create an NQR a such that the Jacobi symbol $(a/n) = +1$ but a is a QR mod p and q. This property would be maintained after re-encryption. To avoid this, every user ensure verify that the modulus n in the public key of the recipient is an RSA modulus, i.e. the product of two primes p and q. There are several techniques available to do so; for example, Graaf and Peralta show how to prove that $n = p^r q^s$ for primes $p, q \equiv 3 \mod 4$ and r and s odd [22] and Boyar *et al.* show how to prove that n is square-free [4]; combining these two proofs yields a proof that $n = pq$. Alternately, Juels and Guajardo present techniques for generating keys with verifiable randomness [14], proving that an RSA public key was constructed using a particular key generation algorithm, and hence is correctly formed. We note that this test need not be performed on each interaction, but only the first time the public key is obtained; indeed, a certificate authority (CA) can require such proofs before issuing certificates.

5. FIXING MK WITH COMMUTATIVE ENCRYPTION

In this section, we describe how to modify the MK algorithm to be secure using the single-bit re-encryption primitive. Single-bit re-encryption eliminates covert channels, but does not support the full range of encryption operations used in the MK algorithm. Namely, MK uses nested encryption ($E_i(E_j(T))$) when sending already encrypted facts to other principals (as seen in Figure 4), which is impossible to represent with single-bit re-encryption. We replace nested encryptions in MK with *commutative encryptions*, an extension of single-bit re-encryption that supports encryption of a bit for a set of principals, $E_S(T)$. For example, $E_i(E_j(T))$ would be changed to $E_{\{i,j\}}(T)$. The commutative encryption has similar semantics to nested encryption, except that the two principals p_i and p_j can perform decryption in any order. Commutative encryption is essentially an n-out-of-n threshold encryption. Using commutative encryption is strictly more powerful than the MK algorithm, but it nevertheless preserves the original safety property of the MK algorithm in Theorem 1.

5.1 Modification of the MK algorithm

We show the modified algorithm based on the commutative encryption model with an example in Figure 6, comparing with the MK algorithm based on the nested encryption model. In the new algorithm, we represent an encryption fact as $(f, E_S(T))$ where S is a set of principals. An encrypted fact $(f, E_S(T))$ is decrypted in any order by the principals in set S, and when all the principals in S perform a decryption operation, a fact f is decrypted; that is, if principal p_i decrypts $(f, E_S(T))$, then it obtains $f, E_{S\setminus\{i\}}(T)$. If $S = \{i\}$, p_i obtains a decrypted fact f.

In Figure 6, there are four principals p_0, p_1, p_2, and p_3, and p_2 and p_3 choose p_0 and p_1 as receiver principals respectively according to their confidentiality policies (i.e., $acl_2(f_1) = \{p_0\}$ and $acl_3(f_2) = \{p_1\}$). MK algorithm fails to derive fact f_0 in principal p_0's knowledge base because principal p_1 cannot perform a decryption operation on the nested encrypted fact $E_0(E_1(T))$. On the other hand, with the commutative encryption model, p_1 performs a decryption operation on $E_{\{0,1\}}$ and obtains $E_{\{0\}}$, which p_0 receives and decrypt.

5.2 Implementing Commutative Encryption

We can use the single-bit re-encryption to build a commutative encryption scheme. First, observe that given an encrypted bit $a = E_i(b)$, we can obtain the encryption of the inverse of that bit $E_i(\neg b)$ by computing $a' = a \cdot x_i \pmod{n_i})$ (where (n_i, x_i) is p_i's public key.)

To obtain $E_{\{i,j\}}(b)$ from $E_i(b)$, we simply form a pair $(E_i(b), E_j(0))$. We can then re-randomize the pair by picking a random bit b' and, if $b' = 1$, negating both elements of the pair to obtain $(E_i(\neg b), E_j(1))$. The pair effectively forms two encrypted shares of the secret bit b [21], so the decryption of both is necessary to recover the original value of b. Similarly, for a larger set $E_S(b)$, there would be a sequence of $|S|$ shares of b, encrypted with the public key of the principals in S.

To re-encrypt a commutative encryption represented as a sequence $(a_1, \ldots, a_n)$, we first re-encrypt each a_i and then we re-randomize the shares: we let $b_1, \ldots, b_{n-1}$ be random, and $b_n = b_1 \oplus \cdots \oplus b_{n-1}$. For each $b_i = 1$, we let $a_i' = \neg a_i$. This step is necessary since otherwise an adversary could use the pattern of the random shares to encode a covert message.

More formally, we can define the following operations:

$\mathsf{Encrypt}(b, p_i) = y^2 x_i^b \pmod{n_i}$ for randomly chosen y.

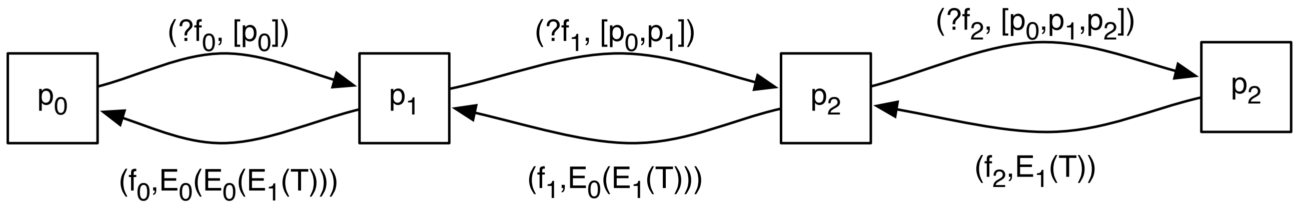
$\mathsf{Encrypt}(E_S(b), p_i) = E_S(b) \cup E_i(0)$. After encryption, the shares should be re-randomized using the $\mathsf{Reencrypt}$ operation.

$\mathsf{Decrypt}(E_i(b), p_i) = b$, where $b = 0$ if $E_i(b)$ is a QR mod n and 1 otherwise.

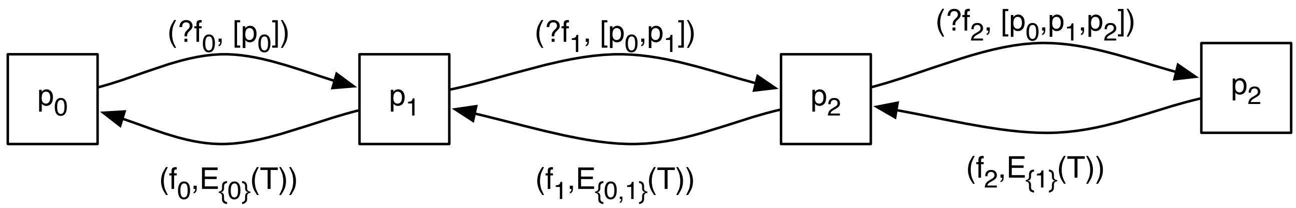
$\mathsf{Decrypt}(E_S(b), p_i) = E_{S-\{i\}}(b)$. If $E_S(b) = (a_j)_{j \in S}$, let $b' = 0$ if a_i is a QR mod n_i and 1 otherwise, and let i' be an element of $S - \{i\}$. Then:

$$E_{S-\{i\}}(b) = (a_j)_{j \in S-\{i,i'\}} \cup (a_{i'} x_{i'}^{b'} \pmod{n_{i'}})$$

Again, $\mathsf{Reencrypt}$ should be used to re-randomize the shares.



(a) Nested encryption model



(b) Commutative encryption model

Figure 6: Comparison of a proof building with nested and commutative encryption models. All the principals have the same knowledge bases, confidentiality policies, and integrity policies in both cases. Principals p_2 and p_3 choose p_0 and p_1 as receiver principals respectively according to their confidentiality policies (i.e., $acl_2(f_1) = \{p_0\}$ and $acl_3(f_2) = \{p_1\}$). Principal p_0 derives fact f_0 in Figure 6(b), but not in Figure 6(a).

$\mathsf{Reencrypt}(E_S(b))$ If $E_S(b) = (a_j)_{j\in S}$, then:

$$\mathsf{Reencrypt}(E_S(b)) = (a_j y_j^2 x_j^{b_j} \pmod{n_j})_{j\in S}$$

with y_j's picked at random and b_j's picked randomly but with the constraint that $\bigoplus_{j\in S} b_j = 0$. Then: $\mathsf{Reencrypt}$ must also check that each a_j is well-formed, using the gcd and Jacobi symbol checks from Section 4.2.

5.3 Safety with commutative encryption model

It is easy to see that Proposition 1 and Lemma 1 in Section 2.4 hold equally well with commutative encryption used in place of nested encryption, since we can construct alternate commutative encryptions in exactly the same way as nested encryptions. Therefore, the new algorithm based on commutative encryption model also satisfies the safety property of the MK algorithm.

6. DISCUSSION

Single bit re-encryption may be useful in other applications. For example, in electronic voting, there may be a need to make sure that a voting machine generates correct values and does not create covert channels as it transmits encrypted votes. Another application may be for online games, where a game server may want to create a limited communication channel between two players. By allowing k single-bit re-encryptions, the parties can be restricted to communicating at most k bits of information.

An alternate approach for restricting the domain of an encrypted domain would be to use zero-knowledge proofs; e.g. Damgard and Jurik develop proof techniques to show that an encrypted value is one of two possibilities [5]. Zero-knowledge proofs, however, contain randomness chosen by the prover, which can itself be used as a covert channel. Our scheme is also significantly more efficient than zero-knowledge proofs.

Quadratic residues have been used in other cryptographic protocols, such as the Kushilevitz–Ostrovsky scheme for private information retrieval [15]. Commutative encryption schemes have been previously developed by Pohlig and Hellman [20] and by Massey and Omura [16]. Weis designed a commutative encryption scheme that is quite similar to ours [23], but used multiplicative rather than additive sharing of the encrypted text. He also proposed definitions of security for commutative encryption.

7. CONCLUSION

We have shown that while a high-level analysis of the MK algorithm shows it to be secure, the implementation details contain a covert channel that leaks confidential information. Our work demonstrates the importance of analyzing cryptographic protocols at the level of implementation, rather than just at a high level, especially when confidentiality is a concern.

We have developed single-bit re-encryption based on the Goldwasser–Micali cryptosystem, and extended it to support commutative/threshold encryption. This primitive is able to remedy the covert channel problem in the MK algorithm, and in fact our improved design is also strictly more powerful than the original algorithm due to more flexible order of decryption. We are also exploring other applications of single-bit re-encryption in the domains of electronic voting and online games; we hope it will be useful as a means of simplifying zero-knowledge proofs or removing covert channels in those applications.

Acknowledgments

We would like to thank Ian Goldberg for some initial discussions about quadratic residues, the anonymous reviewers for their helpful suggestions, and our shepherd, Ben Adida for his help with the final version of the paper. Minami was supported by the Office of Science and Technology at the Department of Homeland Security. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security or the Office of Science and Technology.

8. REFERENCES

[1] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the Grey system. In J. Zhou and J. Lopez, editors, *Information Security Conference*, volume 3650 of *Lecture Notes in Computer Science*, pages 431–445, Singapore, Sept. 2005.

[2] L. Bauer, S. Garriss, and M. K. Reiter. Distributed proving in access-control systems. In V. Paxson and M. Waidner, editors, *IEEE Symposium on Security and Privacy*, pages 81–95, Washington, DC, USA, May 2005. Computer Society.

[3] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Perugia, Italy, May 1994.

[4] J. Boyar, K. Friedl, and C. Lund. Practical zero-knowledge proofs: Giving hints and using deficiencies. *Journal of Cryptology*, 4(3):185–206, Jan. 1991.

[5] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *Workshop on Practice and Theory in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, Korea, Feb. 2001.

[6] J. DeTreville. Binder, a logic-based security language. In M. Abadi and S. M. Bellovin, editors, *IEEE Symposium on Security and Privacy*, pages 105–113, Oakland, CA, USA, May 2002.

[7] S. M. Eikenberry and J. P. Sorenson. Efficient algorithms for computing the Jacobi symbol. *Journal of Symbolic Computation*, 26(4):509–523, 1998.

[8] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In J. Kilian, editor, *Advances in Cryptology (CRYPTO)*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387, Santa Barbara, CA, USA, Aug. 2001.

[9] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In H. R. Lewis, editor, *ACM Symposium on Theory of Computing*, pages 365–377, San Francisco, CA, May 1982.

[10] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[11] P. Golle and D. Boneh. Almost entirely correct mixing with applications to voting. In R. Sandhu, editor, *9th ACM Conference on Communications and Computer Security*, pages 68–77, Washington, DC, Oct. 2002.

[12] M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In D. Boneh, editor, *USENIX Security Symposium*, pages 339–353, San Francisco, CA, Aug. 2002.

[13] T. Jim. SD3: A trust management system with certified evaluation. In R. Needham and M. Abadi, editors, *IEEE Symposium on Security and Privacy*, pages 106–115, Berkeley, CA, May 2001. Society.

[14] A. Juels and J. Guajardo. RSA key generation with verifiable randomness. In D. Naccache and P. Paillier, editors, *Workshop on Practice and Theory in Public Key Cryptosystems*, volume 2274 of *Lecture Notes in Computer Science*, pages 261–285, Paris, France, Feb. 2002.

[15] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Symposium on Foundations of Computer Science*, Miami Beach, FL, Oct. 1997.

[16] J. Massey and J. Omura. A new multiplicative algorithm over finite fields and its applicability in public key cryptography. Presented at the rump session of EUROCRYPT, Mar. 1983.

[17] K. Minami and D. Kotz. Secure context-sensitive authorization. *Journal of Pervasive and Mobile Computing*, 1(1):123–156, Mar. 2005.

[18] K. Minami and D. Kotz. Secure context-sensitive authorization. In K. Nahrstedt, editor, *IEEE International Conference on Pervasive Computing and Communications*, pages 257–268, Kauai, HI, Mar. 2005.

[19] C. A. Neff. A verifiable secret shuffle and its applications to e-voting. In P. Samarati, editor, *8th ACM conference on Computer and Communications Security*, pages 116–125, Philadelphia, PA, Oct. 2001.

[20] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1):106–110, Jan. 1978.

[21] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[22] J. van de Graaf and R. Peralta. A simple and secure way to show the validity of your public key. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 128–134, Santa Barbara, CA, Aug. 1987.

[23] S. A. Weis. *New Foundations for Efficient Authentication, Commutative Cryptography, and Private Disjointness Testing*. PhD thesis, Massachusetts Institute of Technology, May 2006.

[24] M. Winslett, C. C. Zhang, and P. A. Bonatti. PeerAccess: A logic for distributed authorization. In C. Meadows, editor, *12th ACM Conference on Computer and Communications Security*, pages 168–179, Alexandria, VA, USA, 2005.