

Advanced Cryptography
Link between Computational and Symbolic, Introduction to
Cryptoverif and others
1st December, 2008
[SUMMARY]

Link between Computational and Symbolic world

The analysis of security protocols is being carried out mainly by means of two different techniques:

- logical perspective (Symbolic)

On one hand, from a *logical perspective*, messages are seen as algebraic objects, generated by some grammar from elementary objects such as keys, nonces, and constants.

Cryptographic operations are seen as algebraic operations which are unbreakable. Attackers are typically modeled as so-called Dolev-Yao attackers, having total control over the network, no computational limitations, and being only (but absolutely) incapable of breaking cryptographic operations. These logical methods are appealing, because they are relatively easy to use and capture most mistakes commonly made in security protocols.

- complexity-theory perspective (Computational)

On the other hand, from a *complexity-theory perspective*, messages are seen as bit strings and cryptographic operations as functions on bit strings satisfying certain security properties. An attacker here is a resource bounded probabilistic algorithm, limited by running time and/or memory, but capable of breaking cryptographic operations, if that is computationally feasible. The complexity based methods are more general and more realistic, but also more complex.

What is the connection between Computational and Symbolic worlds?

Such a relation takes the form of a function mapping algebraic messages m to (distributions over) bit strings $[m]$. This map should relate messages that are observationally equivalent in the algebraic world (meaning that a Dolev-Yao attacker can see no difference between them) to indistinguishable distributions over bit strings

(meaning that a computationally bounded adversary can only with negligible probability distinguish the distributions).

Such a map allows one to use algebraic methods, possibly even automated, to reason about security properties of protocols and have those reasoning beveled also in the computational world.

Patterns

Abadi and Rogaway define $E_1 = E_2$ as follows:

For a formal expression E , define the set keys K that occur in E but that the adversary cannot find

Replace sub expressions $\{\dots\}_K$ of E , where $k \in K$, by (“the undecryptable”)

- This gives the **pattern** of E
- Denote it by $P(E)$
- $E_1 = E_2$ if $P(E_1) = P(E_2)$

Key Recovery Function $F_{Kr}(E, K)$

The approach of (Micciancio , Warinschi), for an expression E and a set K of Keys , we define Key Recovery Function $F_{Kr}(E, K)$ as follows:

- $F_{Kr}(E, K) = \phi$
- $F_{Kr}(k, K) = \{k\} \cup K$
- $F_{Kr}((E_1, E_2), K) = F_{Kr}(E_1, K) \cup F_{Kr}(E_2, K)$
- $F_{Kr}(\{E_k\}, K) = \begin{cases} K & \text{if } k \in K \\ F_{Kr}(E, K) & \text{otherwise} \end{cases}$

Inductive definition of key recovery (rec)

Let E be an expression, then we define rec by :

- $rec(E) = U_i G_i(E) = G_{|E|}(E)$ where
- $G_0(E) = \phi$
- $G_i(E) = F_{Kr}(E, G_{i-1}(E))$

$rec(E)$ consists of the keys that can be recovered from E using information available in E

The pattern of an expression

We denote by $Keys(E)$ the set of all keys occurring in E , and let $hidden(E) = Keys(E) - rec(E)$. Define $Pat(E, K)$ as follows:

- $Pat(b, K) = b$
- $Pat(k, K) = k$
- $Pat((E_1, E_2), K) = (Pat(E_1, K), Pat(E_2, K))$
- $Pat(\{E\}_k, K) = \begin{cases} Pat(E, K) & \text{if } k \in K \\ \square & \text{otherwise} \end{cases}$

Finally, we define pat by $Pat(E) = Pat(E, rec(E))$. Intuitively, $Pat(E)$ is the pattern corresponding to E , given the knowledge of any keys that may be recovered from E .

Example:

- $E = ((\{k_2\}_{k_1}, \{k_3\}_{k_2}), k_1)$
- $rec(E) = \{k_1, k_2, k_3\}$
- $Pat(E) = E$

Pattern Equivalence

See [power point slides by Pascal Lafourcade ,Link between Computational and symbolic (slide 14)].

Encryption Cycles

There is a definite gap between formal and computational worlds. Key k_1 encrypts key k_2 in expression E , if some $\{\dots k_2 \dots\}_{k_1}$ is a sub expression of E . Here k_2 occurs not only as the encryption key.

An encryption cycle is a cycle of the relation encrypts.

Example: $(\{k_2\}_{k_1}; \{k_1\}_{k_2})$

Encryption cycles are secure in symbolic world, insecure in computational. Abadi and Rogaway define their equivalence relation only for expressions without encryption cycles.

CryptoVerif

CryptoVerif is an automatic protocol prover sound in the computational model. It can prove :

- Secrecy
- Correspondences, which include in particular authentication.

It provides a generic mechanism for specifying the security assumptions on cryptographic primitives, which can handle in particular symmetric encryption, message authentication codes, public-key encryption, signatures, hash functions.

The generated proofs are proofs by sequences of games, as used by cryptographers. These proofs are valid for a number of sessions polynomial in the security parameter, in the presence of an active adversary.

CryptoVerif can also evaluate the probability of success of an attack against the protocol as a function of the probability of breaking each cryptographic primitive and of the number of sessions (exact security).

For more information about CryptoVerif tool; See [power point slides by Pascal Lafourcade ,Link between Computational and symbolic (slide 25-42)].

E-Cash

E-Cash (electronic cash) refers to an anonymous electronic cash system (a way that the money is exchanged electronically and involves the use of computer network, Internet and digital stored value system). Typically, e-cash uses the cryptography and hashing to establish a link between withdrawal and spend transaction. E-Cash could be categorized into two parts. First, on-line payment in which bank confirmation will be required. Second, off-line payment in which the previous confirmation will not be needed. E-Cash

is a sensitive kind of trade that insures and guarantee a mutual trust between all participates in this kind of trade. Therefore, it should provides the following features:

- Confidentially
- Authentication
- Integrity
- Non-repudiation
- Privacy

At the time sequence, e-cash concept has been implemented with a several contributors such as e-cheques1993 by Netchash, EMV 1994 by MasterCard and credit cards over SSL by PayPal (the most popular way for e-cash in US currently). Among the set of protocols available nowadays on Internet that provide a secure electronic transaction, SET and CSET protocols in which they have been implemented and deployed in both VISA and MasterCard in 1997 will be discussed in this report.

Both of these protocols consist of three principles:

- Customer (C)
- Merchant (M)
- Bank (B)

The main idea behind those protocols (among the features mentioned before) is to separate the interests of each principles in which only the required information will be available whence its needed. For instance, the merchant does not need to know about the customer's credit card number. That's goes for the Bank as well in which it does not need to know about the purchase items selected by the customer. The freshness of the transaction is another features guaranteed by these protocols.

SET (Secure Electronic Transaction) protocol uses RSA, DES algorithms and SHA-1 hushing function for the data being transmitted over Internet. As mentioned before about separating the interests, the customer and bank will be the only participants knows about credit card number. The details of selected items will be known by customer and merchant only. The price and quantity of those items will be shared

among the three participants. Its important to introduce a couple of notations that will be used to explain the content of transactions made or delivered between the participants.

SET Protocol Transaction Notations:

- N_M and N_C = Nonces
- O_d = Order Details
- P_d = Payment Details
- $Trans = C, M, (N_C, N_M), Price, hash(O_d), hash(P_d)$

SET Protocol Transaction Contents:

- $C \rightarrow B : C, M$
- $M \rightarrow C : N_M$
- $C \rightarrow M : \{Trans\}_{K_C^{-1}}, \{O_d\}_{K_M}, \{P_d\}_{K_B}$
- $M \rightarrow B : \{Trans\}_{K_C^{-1}}, \{Trans\}_{K_M^{-1}}, \{P_d\}_{K_B}$
- $B \rightarrow M : \{Results, hash(Trans)\}_{K_B^{-1}}$
- $M \rightarrow C : \{Results, hash(Trans)\}_{K_B^{-1}}$

Claimed Properties:

- O_d is a shared secret between C and M
- P_d is a shared secret between C and B
- Mutual authentication between C and M with N_M and N_C
- Authentication of B by C and M wit $hash(Trans)$
- Authentication of all messages by the 3 principles
- Non-repudiation due to signed and hashed $Trans$ by all principles
- Freshness of the transaction checked by B

Not achieved properties

- No anonymity: B knows that C buy something at M
- No Fairness S signs before M

Interactive Zero Knowledge Proof

Interactive zero knowledge proof is an interactive method for one party P (called prover) to prove to another party V (called verifier) the validity of assertion he claimed without revealing any single information about the assertion to the verifier.

The correctness of such proof system by a verifier will require applying two main concepts:

Completeness: if the prover or the person claiming possession of proof provides a valid proof then the verifier will accept the proof (i.e. the proof is accepted).

Soundness: if the prover or the person claiming possession of proof provides invalid proof (does not know the proof) then outputs false (i.e. the proof is rejected).

3-Coloring problem shown in Figure 1 will be used as an example to illustrate the interactive zero knowledge proof concept.

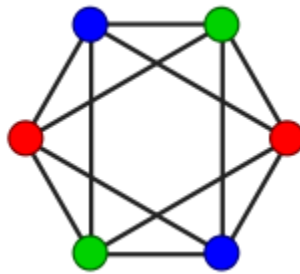


Figure 1: 3-Coloring Graph

3-Coloring problem by itself is a graph in which its vertices called "colors" and subject to certain constraints such that no pair of adjacent vertices will be assigned with the same color. It has been proven that giving such a graph G and decides if it is 3-colorable or not belongs to NP problem.¹

¹ M. R. Garey, D. S. Johnson, and L. Stockmeyer. Proceedings of the sixth annual ACM symposium on Theory of computing, p.47-63. 1974.

In such problem, a person A wants to prove to another person B that he can decide if the given graph G if it is 3-colorable or not without revealing the knowledge used to solve this problem. The proving process will go as the following:

- $A \rightarrow B : \forall u \in V, e_u = H(\pi(c(u)) || r_u)$
- $B \rightarrow A : u, v$
- $A \rightarrow B : r_u, r_v, \pi(c(u)), \pi(c(u)), \pi(c(v))$

Secret Sharing

Secret Sharing: The secret key can be divided into several parts, giving each participant its own unique part, and we are going to need some parts of the keys or all of them to reconstruct the main key.

Example:

We have 11 scientists working on a project and storing the project into a cabinet! That cabinet can be opened if and only if there are 6 or more scientists are present at the same time. We have here: $N = 11$ which is the number of the maximum participants in the project. $M = 6$ which is the number of minimum presented participants needed to open the cabinet.

We have 2 questions here now:

- What are the smallest numbers of locks needed for the cabinet?
- What is the smallest number of keys each participant must carry out?

The Idea for the first question is to have the other 5 different participants groups at least one lock they don't have the key whereas the other participants (group of 6) has the key for. There are $\binom{11}{5} = 462$ groups of 5 participants, so we should have 462 locks.

The second problem idea is to have each participant to hold at least one key for every group of 5 participants of which he is not a member of that group. So there going to be $\binom{105}{5} = 242$ different groups.

The generalized equation for this problem is going to be $\binom{N}{M-1}$ and $\binom{N-1}{M-1}$.

What are the security principles that could be achieved by applying the secret key sharing ?

The two main security that could be achieved by the secret sharing are the data integrity and the data confidentiality, but sharing is also very dangerous if done without being careful. For example if I shared the data with one person for example, this person could sell this data to someone else which violates the idea of confidentiality. The same problem can be happened if the data been shared by many people, it could leak out or be sold to someone, so that data need to be distributed over the participants in the project in a way that if someone gave up his information to another non-authorized party, that party won't be able to get benefit from this information. So, distributing the data can achieve the confidentiality part of the secret sharing concept. But still the data needs to be distributed in a way that can I reconstruct it back again without losing or changing the data otherwise this going to violate the other term of secret sharing which is the data integrity !

Weakness of Shamir-Secret-sharing:

- There is no mechanics to know the shared is valid or not.
- The dealer (key owner) has a total confidence.