Complexity and Cryptography

Thomas Zeugmann

Hokkaido University
Laboratory for Algorithmics

https://www-alg.ist.hokudai.ac.jp/~thomas/COCRB/

Lecture 14: Authentication, Cryptographic Protocols

Motivation I

Question

Why do we need authentication?

Motivation I

Question

Why do we need authentication?

We finished the last lecture by showing how realize the idea to use a bag with two locks. Though our solution looked good, it is not perfect. The problem is a possible attack by a third party in the middle between Alice and Bob. Thus, we need authentication.

Motivation I

Question

Why do we need authentication?

We finished the last lecture by showing how realize the idea to use a bag with two locks. Though our solution looked good, it is not perfect. The problem is a possible attack by a third party in the middle between Alice and Bob. Thus, we need authentication.

Looking again at the bag with two locks, it is easy to see that Alice *must* have a possibility to verify that the lock on the bag is really Bob's lock, since otherwise a third party (Claire) could pretend to be Bob and thus get access to the message in the bag.

Motivation II

This leads us directly to the problem of *authentication*. By *authentication* we mean a method ensuring that a message received indeed originates from the source it pretends to be have sent off.

In classical communication via letters, this problem has been solved by using hand written signatures (in the western hemisphere) or a "hanko," i.e., a personal seal (e.g. in China, Japan). Thus, we need something equivalent, i.e., an electronic signature.

Motivation II

This leads us directly to the problem of *authentication*. By *authentication* we mean a method ensuring that a message received indeed originates from the source it pretends to be have sent off.

In classical communication via letters, this problem has been solved by using hand written signatures (in the western hemisphere) or a "hanko," i.e., a personal seal (e.g. in China, Japan). Thus, we need something equivalent, i.e., an electronic signature.

That is, we are looking for a method that may serve as a *digital signature* but it should be more resistant to forgery than the usually used hand written signatures.

Motivation III

As we shall see, mathematically, it works beautifully. But there are several points of concern which we shortly address here and which must be taken into consideration whenever applications are concerned.

A digital signature authenticates the document by using a *computer*. What is *not* authenticated is the link between that computer and the person for whom authentication is done.

This is a subtle point. It is a matter of faith that my computer is signing the document I intend it to. It is also a matter of faith that my computer is not sending my private key used for signing a document to someone else who can then sign whatever she whishes in my name.

Motivation III

As we shall see, mathematically, it works beautifully. But there are several points of concern which we shortly address here and which must be taken into consideration whenever applications are concerned.

A digital signature authenticates the document by using a *computer*. What is *not* authenticated is the link between that computer and the person for whom authentication is done.

This is a subtle point. It is a matter of faith that my computer is signing the document I intend it to. It is also a matter of faith that my computer is not sending my private key used for signing a document to someone else who can then sign whatever she whishes in my name.

Motivation III

As we shall see, mathematically, it works beautifully. But there are several points of concern which we shortly address here and which must be taken into consideration whenever applications are concerned.

A digital signature authenticates the document by using a *computer*. What is *not* authenticated is the link between that computer and the person for whom authentication is done.

This is a subtle point. It is a matter of faith that my computer is signing the document I intend it to. It is also a matter of faith that my computer is not sending my private key used for signing a document to someone else who can then sign whatever she whishes in my name.

Authentication I

Next, we describe a *principal* mathematical solution. The following method is due to Taher Elgamal (1985) and based on the Diffie-Hellman public key cryptosystem:

Authentication I

Next, we describe a *principal* mathematical solution. The following method is due to Taher Elgamal (1985) and based on the Diffie-Hellman public key cryptosystem:

Recall that q is a huge prime, and that g is a generator of $\mathbb{Z}_q^*$. Furthermore, Bob's public key is y_B.
To send his signature S, Bob chooses a random integer k with $\gcd(k, q-1) = 1$. Then Bob calculates $r = g^k \mod q$, solves the following congruence for the unknown x:

$$g^S \equiv y_B^r r^x \mod q\,, \tag{1}$$

and sends Alice the pair (r, x) along with S.
Now, Alice can verify that $g^S \equiv y_B^r r^x \mod q$, and she is happy, secure in her confidence that Bob did send the message S.

Authentication II

It remains to argue that Bob can perform efficiently all the computations necessary, and that Alice can be really happy.

Authentication II

It remains to argue that Bob can perform efficiently all the computations necessary, and that Alice can be really happy.

Obviously, r and g^s can be efficiently computed by using our modular exponention Algorithm EXP.

Authentication II

It remains to argue that Bob can perform efficiently all the computations necessary, and that Alice can be really happy.

Obviously, r and g^s can be efficiently computed by using our modular exponention Algorithm EXP.

But what about solving $g^s \equiv y_B^r r^x \mod q$?

Authentication II

It remains to argue that Bob can perform efficiently all the computations necessary, and that Alice can be really happy.

Obviously, r and g^S can be efficiently computed by using our modular exponention Algorithm EXP.

But what about solving $g^S \equiv y_B^r r^x \mod q$?

Taking into account that $y_B \equiv g^{x_B} \mod q$, we obtain, by putting it all together:

$$g^S = g^{x_B r} g^{kx} \equiv g^{x_B r + kx} \mod q \tag{2}$$

Thus, by applying Euler's theorem to (2), we obtain the condition

$$S \equiv x_B r + kx \mod (q-1), \tag{3}$$

and hence, $x \equiv (S - x_B r)k^{-1} \mod (q-1)$.

Authentication III

So we see why the number k has been required to satisfy $\gcd(k, q-1) = 1$. Clearly, all the needed computations can be efficiently performed by Bob.

Authentication III

So we see why the number k has been required to satisfy $\gcd(k, q-1) = 1$. Clearly, all the needed computations can be efficiently performed by Bob.

Finally, Alice can be sure to have obtained Bob's signature, since solving the congruence $g^s \equiv y_B^r r^x \mod q$ in order to determine x requires the knowledge of x_B which is kept secretly by Bob. Forging Bob's signature is as complicated as computing discrete logarithms.

Authentication III

So we see why the number k has been required to satisfy $\gcd(k, q-1) = 1$. Clearly, all the needed computations can be efficiently performed by Bob.

Finally, Alice can be sure to have obtained Bob's signature, since solving the congruence $g^s \equiv y_B^r r^x \mod q$ in order to determine x requires the knowledge of x_B which is kept secretly by Bob. Forging Bob's signature is as complicated as computing discrete logarithms.

Now, we also understand why the parties have to publish their key y. It is either for using it for the key exchange described in the previous lecture, or for performing the authentication as described above.

Authentication IV

The only thing that kind of remains open is how S is chosen by Bob.

Using the protocol provided above, the best choice is to use the whole message Bob wishes to send as signature S. One advantage of the protocol described above is the probabilistic element introduced by the random choice of k. On the other hand, there is also a serious disadvantage, since the ciphertext to be send it now blown up.

So, in principal there is a beautiful mathematical solution to the authentication problem. Next, we put authentication in the more general context of cryptographic protocols.

Authentication IV

The only thing that kind of remains open is how S is chosen by Bob.

Using the protocol provided above, the best choice is to use the whole message Bob wishes to send as signature S. One advantage of the protocol described above is the probabilistic element introduced by the random choice of k. On the other hand, there is also a serious disadvantage, since the ciphertext to be send it now blown up.

So, in principal there is a beautiful mathematical solution to the authentication problem. Next, we put authentication in the more general context of cryptographic protocols.

Authentication IV

The only thing that kind of remains open is how S is chosen by Bob.

Using the protocol provided above, the best choice is to use the whole message Bob wishes to send as signature S. One advantage of the protocol described above is the probabilistic element introduced by the random choice of k. On the other hand, there is also a serious disadvantage, since the ciphertext to be send it now blown up.

So, in principal there is a beautiful mathematical solution to the authentication problem. Next, we put authentication in the more general context of cryptographic protocols.

Cryptographic Protocols I

Definition 1

Cryptographic protocols describe algorithms used for the communication between different parties, adversaries or not.

Cryptographic Protocols I

Definition 1

Cryptographic protocols describe algorithms used for the communication between different parties, adversaries or not.

By definition, cryptographic protocols apply cryptographic transformations. Consequently, they are at most as secure as the underlying cryptosystem. Usually, we shall use public key cryptosystems for cryptographic protocols.

Cryptographic Protocols I

Definition 1

Cryptographic protocols describe algorithms used for the communication between different parties, adversaries or not.

By definition, cryptographic protocols apply cryptographic transformations. Consequently, they are at most as secure as the underlying cryptosystem. Usually, we shall use public key cryptosystems for cryptographic protocols.

However, the goal of the protocol is usually something beyond the simple secrecy of message transmission.

Cryptographic Protocols II

For example, the communicating parties may want to share parts of their secrets to achieve a common goal, or they like to convince the other parties that they know a particular secret *without* providing even a single bit of the secret on hand.

Protocols realizing such goals have considerably changed our understanding about what is impossible when several parties, adversaries or not, communicate with each other.

Cryptographic Protocols III

To see how digital signatures fit into the domain of protocols let us consider the following very general task: A *private* conversation should be established between two individual users of an information system or a communication network. We do not make any assumption concerning whether or not these two individual users have ever communicated with each other before.

Having a public key cryptosystem on hand, we can solve this problem. First, our users publish their *public key*. Then, messages send to user A are encrypted by using A's public key.

Cryptographic Protocols III

To see how digital signatures fit into the domain of protocols let us consider the following very general task: A *private* conversation should be established between two individual users of an information system or a communication network. We do not make any assumption concerning whether or not these two individual users have ever communicated with each other before.

Having a public key cryptosystem on hand, we can solve this problem. First, our users publish their *public key*. Then, messages send to user A are encrypted by using A's public key. But even if the cryptosystem is considered to be secure, we still have to deal with the problem that a user C might pretend to be the user B when sending a message to A. To prevent the occurrence of such situations, some convention of *signing* messages has to be added to the protocol.

Cryptographic Protocols IV

One always has to *separate* security properties of the underlying cryptosystem from those of the protocol. When doing this, the possible adversaries should be kept in mind. In most communication protocols, an adversary belongs to one of the following three types:

Cryptographic Protocols IV

One always has to *separate* security properties of the underlying cryptosystem from those of the protocol. When doing this, the possible adversaries should be kept in mind. In most communication protocols, an adversary belongs to one of the following three types:

(1) Communicating parties who try to cheat. Later we shall meet two types of cheaters, i.e., passive and active.

Cryptographic Protocols IV

One always has to *separate* security properties of the underlying cryptosystem from those of the protocol. When doing this, the possible adversaries should be kept in mind. In most communication protocols, an adversary belongs to one of the following three types:

(1) Communicating parties who try to cheat. Later we shall meet two types of cheaters, i.e., passive and active.

(2) Passive eavesdroppers. They may obtain information not intended for them, but are otherwise harmless.

Cryptographic Protocols IV

One always has to *separate* security properties of the underlying cryptosystem from those of the protocol. When doing this, the possible adversaries should be kept in mind. In most communication protocols, an adversary belongs to one of the following three types:

(1) Communicating parties who try to cheat. Later we shall meet two types of cheaters, i.e., passive and active.

(2) Passive eavesdroppers. They may obtain information not intended for them, but are otherwise harmless.

(3) Active eavesdroppers. Besides obtaining secret information (as passive eavesdroppers do), they may mess up the whole protocol.

Cryptographic Protocols V

In our problem above, where C tries to impersonate B, we have an adversary of Type (3), i.e., an active eavesdropper. To have an example for Type (1), just imagine that some people like to play poker by telephone. Clearly, somebody might be tempted to cheat. We shall come back to this point below. Looking at typical applications of cryptography, it should be also clear that adversaries of Type (2) may cause huge trouble, e.g., in military or diplomatic applications, or in banking.

Cryptographic Protocols V

In our problem above, where C tries to impersonate B, we have an adversary of Type (3), i.e., an active eavesdropper. To have an example for Type (1), just imagine that some people like to play poker by telephone. Clearly, somebody might be tempted to cheat. We shall come back to this point below. Looking at typical applications of cryptography, it should be also clear that adversaries of Type (2) may cause huge trouble, e.g., in military or diplomatic applications, or in banking.

For the sake of illustrating the difference between an active and passive eavesdropper let us look at the RSA cryptosystem. We claim that it vulnerable against attacks with chosen ciphertext. This can be seen as follows:

Cryptographic Protocols VI

Suppose an eavesdropper E has received

$$c = m^e \bmod n\,. \tag{4}$$

The eavesdropper E wishes to know m. Let A be the legal receiver of m. Clearly, A will not decrypt c for E.

Cryptographic Protocols VI

Suppose an eavesdropper E has received

$$c = m^e \bmod n\,. \tag{4}$$

The eavesdropper E wishes to know m. Let A be the legal receiver of m. Clearly, A will not decrypt c for E. But E can modify c by using a randomly chosen $x \in \mathbb{Z}_n^*$ and computing

$$\hat{c} = cx^e \equiv \hat{m}^e \bmod n\,. \tag{5}$$

Then, the eavesdropper E can send $\hat{c}$ to A.

Cryptographic Protocols VI

Suppose an eavesdropper E has received

$$c = m^e \bmod n\,. \tag{4}$$

The eavesdropper E wishes to know m. Let A be the legal receiver of m. Clearly, A will not decrypt c for E. But E can modify c by using a randomly chosen $x \in \mathbb{Z}_n^*$ and computing

$$\hat{c} = cx^e \equiv \hat{m}^e \bmod n\,. \tag{5}$$

Then, the eavesdropper E can send $\hat{c}$ to A.
If E succeeds to get A to decrypt this message for him, then E gets $\hat{m}$, and thus he knows the original message m, too, since

$$\hat{c} = cx^e \bmod n = m^e x^e \bmod n = (mx)^e \bmod n\,. \tag{6}$$

By construction, $\hat{m} = mx \bmod n$, thus $m \equiv \hat{m}x^{-1} \bmod n$.

Cryptographic Protocols VII

In the preceding example we have not said how E succeeds to get A to decrypt $\hat{c}$ for him.

Cryptographic Protocols VII

In the preceding example we have not said how E succeeds to get A to decrypt $\hat{c}$ for him.

Assuming A is unexperienced, E may have pretended to be B (the original sender). Then, after having encrypted $\hat{c}$, the legal receiver A was confused, since $\hat{m}$ did not make any sense to her. Thus, A sent $\hat{m}$ back to E instead of sending it to B. As we have seen, this is very *dangerous* and should be *avoided at all*.

Cryptographic Protocols VII

In the preceding example we have not said how E succeeds to get A to decrypt $\hat{c}$ for him.

Assuming A is unexperienced, E may have pretended to be B (the original sender). Then, after having encrypted $\hat{c}$, the legal receiver A was confused, since $\hat{m}$ did not make any sense to her. Thus, A sent $\hat{m}$ back to E instead of sending it to B. As we have seen, this is very *dangerous* and should be *avoided at all*.

A much better way to recover from such a confusion is to send a request to B to resend the message.

Cryptographic Protocols VII

In the preceding example we have not said how E succeeds to get A to decrypt $\hat{c}$ for him.

Assuming A is unexperienced, E may have pretended to be B (the original sender). Then, after having encrypted $\hat{c}$, the legal receiver A was confused, since $\hat{m}$ did not make any sense to her. Thus, A sent $\hat{m}$ back to E instead of sending it to B. As we have seen, this is very *dangerous* and should be *avoided at all*.

A much better way to recover from such a confusion is to send a request to B to resend the message.

To make the need of protocols more transparent, we provide two more examples of weak points that may occur when using plain RSA which also apply *mutatis mutandis* to many other plain public key cryptosystems.

Buy, Sell, or Hold

Suppose Alice wants to send orders to her stock broker Bob. An eavesdropper would like to know Alice's order. Furthermore, suppose the eavesdropper has good reason to belive that m is one of the following three messages:

- $m_1 =$ "buy IBM"
- $m_2 =$ "sell IBM"
- $m_3 =$ "hold IBM"

Buy, Sell, or Hold

Suppose Alice wants to send orders to her stock broker Bob. An eavesdropper would like to know Alice's order. Furthermore, suppose the eavesdropper has good reason to belive that m is one of the following three messages:

- $m_1 =$ "buy IBM"
- $m_2 =$ "sell IBM"
- $m_3 =$ "hold IBM"

The eavesdropper can compute the encryptions c_1, c_2, and c_3 of the three messages for himself, and when Alice is sending an encryption of one of these three messages, say m_2, the eavesdropper simple compares the ciphers and knows it is m_2. This example shows that plain RSA can *leak partial information*.

Making the Lowest Bid

Suppose Alice wants to submit a number m, representing her bid, to Bob. Bob is accepting many bids, and will choose the lowest bid.

Making the Lowest Bid

Suppose Alice wants to submit a number m, representing her bid, to Bob. Bob is accepting many bids, and will choose the lowest bid.

Suppose the eavesdropper is a competitor, too, and wants to underbid Alice by 10%. We make the reasonable assumption that Alice's bid is made in round numbers and thus amounts to be a multiple of 10. Then the eavesdropper can intercept Alice's *encrypted* message c. Now, he computes

$$\hat{c} = c \cdot \left(9 \cdot 10^{-1}\right)^{e} \bmod n\,, \tag{7}$$

where 10^{-1} is the modular inverse of 10 modulo n. This inverse exists, since n is the product of two large primes and $10 = 2 \cdot 5$. Hence $\gcd(10, n) = 1$. So, we have $\hat{m} = 0.9 \cdot m$. In this way, Alice's competitor can underbid Alice by 10%, *without* knowing anything about the value of Alice's bid.

Cryptographic Protocols VIII

The property observed above that flipping some bits in the ciphertext will also flip some bits in the message is a weakness usually called *malleability* which should *not occur*.

Cryptographic Protocols VIII

The property observed above that flipping some bits in the ciphertext will also flip some bits in the message is a weakness usually called *malleability* which should *not occur*.

More generally speaking, encryption is often identified with "secure envelope" or a locked box that cannot be opened without destroying it.

Cryptographic Protocols VIII

The property observed above that flipping some bits in the ciphertext will also flip some bits in the message is a weakness usually called *malleability* which should *not occur*.

More generally speaking, encryption is often identified with "secure envelope" or a locked box that cannot be opened without destroying it.

This metaphor has a very compelling and convenient touch and is often used by engineers. However, an encryption scheme can at best approximate a "secure envelope."

Cryptographic Protocols VIII

The property observed above that flipping some bits in the ciphertext will also flip some bits in the message is a weakness usually called *malleability* which should *not occur*.

More generally speaking, encryption is often identified with "secure envelope" or a locked box that cannot be opened without destroying it.

This metaphor has a very compelling and convenient touch and is often used by engineers. However, an encryption scheme can at best approximate a "secure envelope."

Question

Why can we at best only approximate a "secure envelope?"

Answers I

Ciphertexts are bit strings (electronically represented) and not physical envelopes.

This has several consequences:

- First, the bit string representing a ciphertext can be *observed* by an eavesdropper. An ideal "secure envelope" leaks no information about the message it contains. For example, if Alice sends two messages to Bob by using a "secure envelope," an eavesdropper cannot tell whether or not these messages are identical or not. The same should hold then for an encryption scheme. But this requirement alone rules out any *deterministic* encryption scheme, i.e., encryption schemes that encrypt the same message always in the same way.

Answers I

Ciphertexts are bit strings (electronically represented) and not physical envelopes.

This has several consequences:

- First, the bit string representing a ciphertext can be *observed* by an eavesdropper. An ideal "secure envelope" leaks no information about the message it contains. For example, if Alice sends two messages to Bob by using a "secure envelope," an eavesdropper cannot tell whether or not these messages are identical or not. The same should hold then for an encryption scheme. But this requirement alone rules out any *deterministic* encryption scheme, i.e., encryption schemes that encrypt the same message always in the same way.

Answers II

- Second, ciphertexts can easily be *replicated*, whereas messages contained in "secure envelopes" cannot. There is really nothing we can do about this. Thus, higher level protocols using encryption must deal with the fact that this can happen.
- Third, ciphertexts can easily be *modified*, creating other ciphertexts as we have seen above. We can do many things to a ciphertext such as flipping some bits from '1' to '0' or vice versa. Even if an encryption scheme is secure, as we have seen, flipping bits in the ciphertext may flip bits in the message. Using the terminology introduced above, we see that malleability cannot be tolerated in many applications. Obviously, malleability has no counterpart in the world of ideal "secure envelopes."

Answers II

- Second, ciphertexts can easily be *replicated*, whereas messages contained in "secure envelopes" cannot. There is really nothing we can do about this. Thus, higher level protocols using encryption must deal with the fact that this can happen.
- Third, ciphertexts can easily be *modified*, creating other ciphertexts as we have seen above. We can do many things to a ciphertext such as flipping some bits from '1' to '0' or vice versa. Even if an encryption scheme is secure, as we have seen, flipping bits in the ciphertext may flip bits in the message. Using the terminology introduced above, we see that malleability cannot be tolerated in many applications. Obviously, malleability has no counterpart in the world of ideal "secure envelopes."

Answers III and a New Question

- Fourth, any bit string is potentially a ciphertext, i.e., the encryption of some message. As we have seen, the fact can also be misused by an adversary who actively participates in a protocol by sending its own messages to other parties. Such a *chosen ciphertext* attack has also no counterpart in the world of ideal "secure envelopes."

Question

Can we overcome these difficulties?

Answers III and a New Question

- Fourth, any bit string is potentially a ciphertext, i.e., the encryption of some message. As we have seen, the fact can also be misused by an adversary who actively participates in a protocol by sending its own messages to other parties. Such a *chosen ciphertext* attack has also no counterpart in the world of ideal "secure envelopes."

Question

Can we overcome these difficulties?

Cryptographic Protocols IX

Hopefully, but it is very difficult to prove something.
For example, when thinking about avoiding attacks from a man in the middle, one can start from the idea that a receiver should acknowledge the receipt of the encrypted message. Thus, if a third party has pretended to be A and has send a message to B, but B is acknowledging it to A, A can immediately inform B that something is wrong.

Cryptographic Protocols IX

Hopefully, but it is very difficult to prove something.
For example, when thinking about avoiding attacks from a man in the middle, one can start from the idea that a receiver should acknowledge the receipt of the encrypted message. Thus, if a third party has pretended to be A and has send a message to B, but B is acknowledging it to A, A can immediately inform B that something is wrong.

Suppose $E_A, E_B, E_C, \ldots$ are the public encryption algorithms of parties $A, B, C, \ldots$ and $D_A, D_B, D_C, \ldots$ are the decryption algorithms kept secretly by $A, B, C, \ldots$. Furthermore, let the following protocol be agreed upon:

Cryptographic Protocols X

For sending a message w from A to B the following steps have to be performed:

(1) A sends the triple $(A, E_B(w), B)$ to B.

(2) B deciphers w by using D_B and sends the triple $(B, E_A(w), A)$ back to A.

Cryptographic Protocols X

For sending a message w from A to B the following steps have to be performed:

(1) A sends the triple $(A, E_B(w), B)$ to B.

(2) B deciphers w by using D_B and sends the triple $(B, E_A(w), A)$ back to A.

At first glance, this protocol looks well designed. But there are some dangers with it.

Let C be an active eavesdropper who has caught the message for B. Since he knows the structure, he changes the triple $(A, E_B(w), B)$ to $(C, E_B(w), B)$ and sends it to B. Following the protocol, B returns the message $(B, E_C(w), C)$ to C. Consequently, C can decipher it and knows w. Since A is waiting for the acknowledgment, she informs B that it did not arrive. Now, B and A realize that something went wrong, but it is too late. Party C already does possess w.

Cryptographic Protocols XI

A better variant might be the following *Challenge-Response* protocol:

Assumption: A and B have a common secret key k and have agreed to use the cryptosystem f.

Cryptographic Protocols XI

A better variant might be the following *Challenge-Response* protocol:
Assumption: A and B have a common secret key k and have agreed to use the cryptosystem f.

A is communicating with someone from whom she expects it is B. For verifying this, the following protocol is used:

The Challenge-Response Protocol

(1) A randomly generates a number r and sends it to B.

(2) The communication partner (hopefully B) encrypts r by using the secret key k in the cryptosystem f and sends $f(r, k)$ back to A.

(3) A computes $f(r, k)$ by herself and compares the computed value with the received one. If they are identical, A assumes that she is indeed communicating with B. If the values are not equal, A concludes that her partner is *not* B.

The Challenge-Response Protocol

(1) A randomly generates a number r and sends it to B.

(2) The communication partner (hopefully B) encrypts r by using the secret key k in the cryptosystem f and sends $f(r, k)$ back to A.

(3) A computes $f(r, k)$ by herself and compares the computed value with the received one. If they are identical, A assumes that she is indeed communicating with B. If the values are not equal, A concludes that her partner is *not* B.

Please think about this protocol and its security.

Playing Poker per Telephone I

First, we have to think about the demands that such a protocol should fulfill.

Playing Poker per Telephone I

First, we have to think about the demands that such a protocol should fulfill.

(i) All hands (sets of five cards) are equally likely.

Playing Poker per Telephone I

First, we have to think about the demands that such a protocol should fulfill.

(i) All hands (sets of five cards) are equally likely.

(ii) The hands of player A and B are disjoint.

Playing Poker per Telephone I

First, we have to think about the demands that such a protocol should fulfill.

(i) All hands (sets of five cards) are equally likely.

(ii) The hands of player A and B are disjoint.

(iii) Both players know their own cards but have no information about the opponent's hand.

Playing Poker per Telephone I

First, we have to think about the demands that such a protocol should fulfill.

(i) All hands (sets of five cards) are equally likely.

(ii) The hands of player A and B are disjoint.

(iii) Both players know their own cards but have no information about the opponent's hand.

(iv) It is possible for each of the players to find out the eventual cheating of the other player.

Playing Poker per Telephone I

First, we have to think about the demands that such a protocol should fulfill.

(i) All hands (sets of five cards) are equally likely.

(ii) The hands of player A and B are disjoint.

(iii) Both players know their own cards but have no information about the opponent's hand.

(iv) It is possible for each of the players to find out the eventual cheating of the other player.

We do not claim this list to be exhaustive.

Playing Poker per Telephone II

Next, we propose a protocol. A cryptosystem, classical or public-key is used. However, neither the encryption methods E_A and E_B nor the decryption methods D_A and D_B are publicized. Furthermore, we assume *commutativity* in any composition of E's and D's. The mutual order is immaterial.

Playing Poker per Telephone II

Next, we propose a protocol. A cryptosystem, classical or public-key is used. However, neither the encryption methods E_A and E_B nor the decryption methods D_A and D_B are publicized. Furthermore, we assume *commutativity* in any composition of E's and D's. The mutual order is immaterial.

Before the actual play, both players A and B agree about the names $w_1, \ldots, w_{52}$ of the 52 cards. The names are chosen in a way such that the cryptosystem is applicable in the sense needed in the sequel. For instance, if E_A and E_B operate on integers in a certain range then each w_i, $i = 1, \ldots, 52$, should be an integer in this range.

Playing Poker per Telephone II

Next, we propose a protocol. A cryptosystem, classical or public-key is used. However, neither the encryption methods E_A and E_B nor the decryption methods D_A and D_B are publicized. Furthermore, we assume *commutativity* in any composition of E's and D's. The mutual order is immaterial.

Before the actual play, both players A and B agree about the names $w_1, \ldots, w_{52}$ of the 52 cards. The names are chosen in a way such that the cryptosystem is applicable in the sense needed in the sequel. For instance, if E_A and E_B operate on integers in a certain range then each w_i, $i = 1, \ldots, 52$, should be an integer in this range.

Player A acts as the dealer but the roles of A and B can be interchanged.

Protocol Poker

Step 1: B shuffles the cards, encrypts them using E_B, and sends them to A, i.e., A receives a random permutation of $E_B(w_1), \ldots, E_B(w_{52})$.
(* A can now only verify that all cards are in the game *)

Step 2: A chooses 5 cards from the sequence received at random and sends them back to B as they are. (* this is B's hand. *) A also encrypts them by using E_A and sends them to B. (* B can check *)

Step 3: A again chooses 5 cards from the remaining cards and encrypts them by using E_A. The result is sent to B, i.e., B receives $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$. (* this is A's hand. *)

Step 4: B applies to these five cards $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$, its own deciphering algorithm D_B and sends the result back to A, i.e., A receives $D_B(E_A(E_B(w_{i_j}))) = E_A(D_B(E_B(w_{i_j}))) = E_A(w_{i_j})$.
(* that is the point where we need commutativity *)

Step 5: Player A applies its own deciphering algorithm D_A to the five cards $E_A(w_{i_j})$. Now, he also nows his hand and the game starts.

Protocol Poker

Step 1: B shuffles the cards, encrypts them using E_B, and sends them to A, i.e., A receives a random permutation of $E_B(w_1), \ldots, E_B(w_{52})$. (* A can now only verify that all cards are in the game *)

Step 2: A chooses 5 cards from the sequence received at random and sends them back to B as they are. (* this is B's hand. *) A also encrypts them by using E_A and sends them to B. (* B can check *)

Step 3: A again chooses 5 cards from the remaining cards and encrypts them by using E_A. The result is sent to B, i.e., B receives $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$. (* this is A's hand. *)

Step 4: B applies to these five cards $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$, its own deciphering algorithm D_B and sends the result back to A, i.e., A receives $D_B(E_A(E_B(w_{i_j}))) = E_A(D_B(E_B(w_{i_j}))) = E_A(w_{i_j})$.
(* that is the point where we need commutativity *)

Step 5: Player A applies its own deciphering algorithm D_A to the five cards $E_A(w_{i_j})$. Now, he also nows his hand and the game starts.

Protocol Poker

Step 1: B shuffles the cards, encrypts them using E_B, and sends them to A, i.e., A receives a random permutation of $E_B(w_1), \ldots, E_B(w_{52})$. (* A can now only verify that all cards are in the game *)

Step 2: A chooses 5 cards from the sequence received at random and sends them back to B as they are. (* this is B's hand. *) A also encrypts them by using E_A and sends them to B. (* B can check *)

Step 3: A again chooses 5 cards from the remaining cards and encrypts them by using E_A. The result is sent to B, i.e., B receives $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$. (* this is A's hand. *)

Step 4: B applies to these five cards $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$, its own deciphering algorithm D_B and sends the result back to A, i.e., A receives $D_B(E_A(E_B(w_{i_j}))) = E_A(D_B(E_B(w_{i_j}))) = E_A(w_{i_j})$.
(* that is the point where we need commutativity *)

Step 5: Player A applies its own deciphering algorithm D_A to the five cards $E_A(w_{i_j})$. Now, he also nows his hand and the game starts.

Protocol Poker

Step 1: B shuffles the cards, encrypts them using E_B, and sends them to A, i.e., A receives a random permutation of $E_B(w_1), \ldots, E_B(w_{52})$. (* A can now only verify that all cards are in the game *)

Step 2: A chooses 5 cards from the sequence received at random and sends them back to B as they are. (* this is B's hand. *) A also encrypts them by using E_A and sends them to B. (* B can check *)

Step 3: A again chooses 5 cards from the remaining cards and encrypts them by using E_A. The result is sent to B, i.e., B receives $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$. (* this is A's hand. *)

Step 4: B applies to these five cards $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$, its own deciphering algorithm D_B and sends the result back to A, i.e., A receives $D_B(E_A(E_B(w_{i_j}))) = E_A(D_B(E_B(w_{i_j}))) = E_A(w_{i_j})$.

(* that is the point where we need commutativity *)

Step 5: Player A applies its own deciphering algorithm D_A to the five cards $E_A(w_{i_j})$. Now, he also nows his hand and the game starts.

Protocol Poker

Step 1: B shuffles the cards, encrypts them using E_B, and sends them to A, i.e., A receives a random permutation of $E_B(w_1), \ldots, E_B(w_{52})$. (* A can now only verify that all cards are in the game *)

Step 2: A chooses 5 cards from the sequence received at random and sends them back to B as they are. (* this is B's hand. *) A also encrypts them by using E_A and sends them to B. (* B can check *)

Step 3: A again chooses 5 cards from the remaining cards and encrypts them by using E_A. The result is sent to B, i.e., B receives $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$. (* this is A's hand. *)

Step 4: B applies to these five cards $E_A(E_B(w_{i_j}))$, $j = 1, \ldots, 5$, its own deciphering algorithm D_B and sends the result back to A, i.e., A receives $D_B(E_A(E_B(w_{i_j}))) = E_A(D_B(E_B(w_{i_j}))) = E_A(w_{i_j})$.

(* that is the point where we need commutativity *)

Step 5: Player A applies its own deciphering algorithm D_A to the five cards $E_A(w_{i_j})$. Now, he also nows his hand and the game starts.

Playing Poker per Telephone III

Let us now see how Requirements (i) through (iv) are fulfilled.

Playing Poker per Telephone III

Let us now see how Requirements (i) through (iv) are fulfilled.

Both players know their own hand and the hands are disjoint, since B can check that the items given in Step 3 are different from those received in Step 2.

Playing Poker per Telephone III

Let us now see how Requirements (i) through (iv) are fulfilled.

Both players know their own hand and the hands are disjoint, since B can check that the items given in Step 3 are different from those received in Step 2.

No conclusive evidence can be presented concerning the remaining Requirements from (i) through (iv). The matter largely depends on how truly one-way functions have been chosen for the encryption algorithms E_A and E_B. For example, it might be impossible to find w_i on the basis of $E_B(w_i)$ but, still, some partial information about w_i could be found.

Playing Poker per Telephone III

Let us now see how Requirements (i) through (iv) are fulfilled.

Both players know their own hand and the hands are disjoint, since B can check that the items given in Step 3 are different from those received in Step 2.

No conclusive evidence can be presented concerning the remaining Requirements from (i) through (iv). The matter largely depends on how truly one-way functions have been chosen for the encryption algorithms E_A and E_B. For example, it might be impossible to find w_i on the basis of $E_B(w_i)$ but, still, some partial information about w_i could be found.

These reflections also show why all algorithms E_A and E_B as well as D_A and D_B must be kept secretly. Otherwise, A could also compute $E_B(w_1), \ldots, E_B(w_{52})$ and would have perfect knowledge about the cards.

Playing Poker per Telephone IV

We can also derive a conclusion concerning the plaintext space of any public-key cryptosystem. It must be so huge that no one can encrypt the possible plaintexts in advance and can perform decryption by simply searching through all resulting ciphertexts.

For further illustration of the difficulties to prove that our requirements are fulfilled, let us consider a more concrete scenario.

Example

Let us assume that A and B have agreed about a huge prime p and to represent the cards as numbers chosen from $\{2, \ldots, p-1\}$. Each player chooses secretly for himself an encryption and decryption exponent e_A, d_A and e_B, d_B, respectively, such that

$$e_A \cdot d_A \equiv e_B \cdot d_B \equiv 1 \mod p-1 .$$

Then encryption and decryption are done in an RSA like fashion, i.e., $E_I(w) = w^{e_I} \mod p$ for $I = A, B$ and decryption of a cipher c is done by computing $D_I(w) = c^{d_I} \mod p$ for $I = A, B$.

Example - Continued

We can prove the following claim:

Claim. *The property to be or not to be a quadratic residue is inherited when using this type of encryption.*

Example - Continued

We can prove the following claim:

Claim. *The property to be or not to be a quadratic residue is inherited when using this type of encryption.*

Proof. Let w be a quadratic residue modulo p. Then we have $\left(\frac{w}{p}\right) = 1$, where $\left(\frac{w}{p}\right)$ denotes the Legendre symbol. By the theorem of Euler, we then know

$$\left(\frac{w}{p}\right) \equiv w^{\frac{p-1}{2}} \equiv 1 \mod p\,.$$

Example - Continued

We can prove the following claim:

Claim. *The property to be or not to be a quadratic residue is inherited when using this type of encryption.*

Proof. Let w be a quadratic residue modulo p. Then we have $\left(\frac{w}{p}\right) = 1$, where $\left(\frac{w}{p}\right)$ denotes the Legendre symbol. By the theorem of Euler, we then know

$$\left(\frac{w}{p}\right) \equiv w^{\frac{p-1}{2}} \equiv 1 \mod p .$$

Therefore, we also have

$$\left(\frac{w^{e_A}}{p}\right) \equiv (w^{e_A})^{\frac{p-1}{2}} \equiv \left(w^{\frac{p-1}{2}}\right)^{e_A} \equiv 1^{e_A} \equiv 1 \mod p .$$

That is, w^{e_A} is a quadratic residue modulo p if and only if w is a quadratic residue modulo p. ∎

Conclusion

If one player has discovered this property she can cheat the other player. For example, the numerical values of the four aces may be all a quadratic residue modulo p. When using the protocol above, clearly A will never send a quadratic residue modulo p to B. Again, the hands are no longer equally likely and (iii) is also violated.

This simple example shows that one cannot take too much care. It is very complicated to prove non-trivial theorems about the security of protocols.

Thank you!

Taher Elgamal