

0x1A Great Papers in Computer Security

Vitaly Shmatikov

<http://www.cs.utexas.edu/~shmat/courses/cs380s/>

W. Diffie and M. Hellman

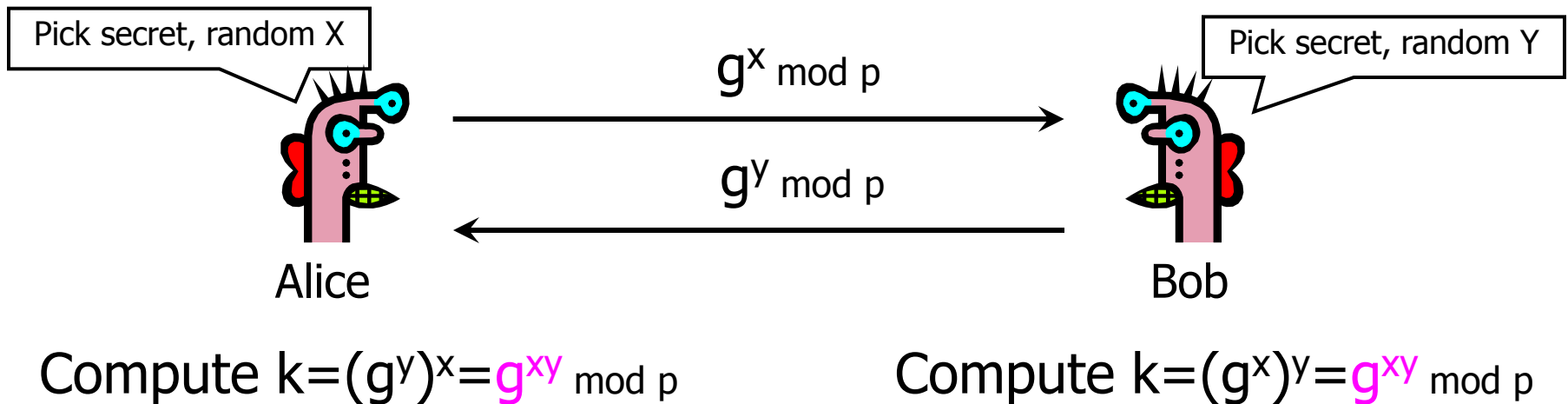
New Directions in Cryptography

(ToIT 1976)



Diffie-Hellman Key Establishment

- ◆ Alice and Bob never met and share no secrets
- ◆ Public information: p and g , where p is a large prime number, g is a generator of Z_p^*
 - $Z_p^* = \{1, 2 \dots p-1\}$; $\forall a \in Z_p^* \exists i$ such that $a = g^i \pmod p$



Why Is Diffie-Hellman Secure?

- ◆ **Discrete Logarithm (DL)** problem:
given $g^x \pmod p$, it's hard to extract x
 - There is no known efficient algorithm for doing this
 - This is not enough for Diffie-Hellman to be secure!
- ◆ **Computational Diffie-Hellman (CDH)** problem:
given g^x and g^y , it's hard to compute $g^{xy} \pmod p$
 - ... unless you know x or y , in which case it's easy
- ◆ **Decisional Diffie-Hellman (DDH)** problem:
given g^x and g^y , it's hard to tell the difference between $g^{xy} \pmod p$ and $g^r \pmod p$ where r is random

Security of Diffie-Hellman Protocol

- ◆ Assuming the DDH problem is hard, Diffie-Hellman protocol is a secure key establishment protocol against passive attackers
 - Eavesdropper can't tell the difference between the established key and a random value
 - Can use the established key for symmetric cryptography
 - Approx. 1000 times faster than modular exponentiation
- ◆ Basic Diffie-Hellman protocol is not secure against an active, man-in-the-middle attacker

Public-Key Encryption

- ◆ **Key generation:** computationally easy to generate a pair (public key PK, private key SK)
 - Computationally infeasible to determine private key SK given only public key PK
- ◆ **Encryption:** given plaintext M and public key PK, easy to compute ciphertext $C = E_{PK}(M)$
- ◆ **Decryption:** given ciphertext $C = E_{PK}(M)$ and private key SK, easy to compute plaintext M
 - Infeasible to compute M from C without SK
 - Trapdoor function: $\text{Decrypt}(SK, \text{Encrypt}(PK, M)) = M$

ElGamal Encryption

◆ Key generation

- Pick a large prime p , generator g of Z_p^*
- Private key: random x such that $1 \leq x \leq p-2$
- Public key: $(p, g, \gamma = g^x \bmod p)$

◆ Encryption

- Pick random k , $1 \leq k \leq p-2$
- $E(m) = (g^k \bmod p, m \cdot \gamma^k \bmod p) = (\gamma, \delta)$

◆ Decryption

- Given ciphertext (γ, δ) , compute $\gamma^{-x} \bmod p$
- Recover $m = \delta \cdot (\gamma^{-x}) \bmod p$

When Is Encryption “Secure”?

- ◆ Hard to recover the key?
 - What if attacker can learn plaintext without learning the key?
- ◆ Hard to recover plaintext from ciphertext?
 - What if attacker learns some bits or some property of the plaintext?
- ◆ (Informal) goal: ciphertext should hide all “useful” information about the plaintext
 - ... except its length

Attack Models

Assume that the attacker knows the encryption algorithm and wants to decrypt some ciphertext

- ◆ Ciphertext-only attack
- ◆ Known-plaintext attack (stronger)
 - Knows some plaintext-ciphertext pairs
- ◆ Chosen-plaintext attack (even stronger)
 - Can obtain ciphertext for any plaintext of his choice
- ◆ Chosen-ciphertext attack (very strong)
 - Can decrypt any ciphertext except the target

The Chosen-Plaintext (CPA) Game

- ◆ Attacker chooses as many plaintexts as he wants and learns the corresponding ciphertexts
- ◆ When ready, he picks two plaintexts M_0 and M_1
 - He is even allowed to pick plaintexts for which he previously learned ciphertexts!
- ◆ He receives either a ciphertext of M_0 , or a ciphertext of M_1
- ◆ He wins if he guesses correctly which one it is

CPA Game: Formalization

- ◆ Idea: attacker should not be able to learn any property of the encrypted plaintext
- ◆ Define $\text{Enc}(M_0, M_1, \mathbf{b})$ to be a function that returns encrypted $M_{\mathbf{b}}$
 - Think of Enc as a magic box that computes ciphertexts on attacker's demand... he can obtain a ciphertext of any plaintext M by submitting $M_0 = M_1 = M$, or he can try to learn even more by submitting $M_0 \neq M_1$
- ◆ Attacker's goal is to learn just one bit \mathbf{b}

0 or 1

Chosen-Plaintext Security

◆ Consider two experiments (A is the attacker)

Experiment 0

A interacts with $\text{Enc}(-,-,0)$
and outputs bit d

Experiment 1

A interacts with $\text{Enc}(-,-,1)$
and outputs bit d

- Identical except for the value of the secret bit
- d is attacker's guess of the secret bit

◆ Attacker's advantage is defined as

$$| \text{Prob}(A \text{ outputs } 1 \text{ in Exp0}) - \text{Prob}(A \text{ outputs } 1 \text{ in Exp1}) |$$

If A "knows" secret bit, he should be able to make his output depend on it

◆ Encryption scheme is chosen-plaintext secure if this advantage is negligible for any efficient A

Simple Example

- ◆ Any deterministic, stateless encryption scheme is insecure against chosen-plaintext attack
 - Attacker can easily distinguish encryptions of different plaintexts from encryptions of identical plaintexts

Attacker A interacts with $\text{Enc}(-,-,b)$

Let X, Y be any two different plaintexts

$C_1 \leftarrow \text{Enc}(X, Y, b);$

$C_2 \leftarrow \text{Enc}(Y, Y, b);$

If $C_1 = C_2$ then output 1 else output 0

- ◆ The advantage of this attacker A is 1

$\text{Prob}(A \text{ outputs } 1 \text{ if } b=0)=0$ $\text{Prob}(A \text{ outputs } 1 \text{ if } b=1)=1$

Semantic Security

[Goldwasser and Micali 1982]

◆ Ciphertext hides even partial information about the plaintext

- No matter what prior knowledge attacker has about the plaintext, it does not increase after observing ciphertext

◆ Equivalent to **ciphertext indistinguishability** under the chosen-plaintext attack

- It is infeasible to find two messages whose encryptions can be distinguished



Semantic Security of ElGamal

Semantic security of ElGamal encryption is equivalent to DDH

- ◆ Given an oracle for breaking DDH, show that we can find two messages whose ElGamal ciphertexts can be distinguished
- ◆ Given an oracle for distinguishing ElGamal ciphertexts, show that we can break DDH
 - Break DDH = given a triplet $\langle g^a, g^b, Z \rangle$, we can decide whether $Z = g^{ab} \pmod p$ or Z is random

DDH \Rightarrow ElGamal

- ◆ Pick any two messages m_0, m_1
- ◆ Receive $E(m) = g^k, m \cdot y^k$
 - $y = g^x$ is the ElGamal public key
 - To break ElGamal, must determine if $m=m_0$ or $m=m_1$
- ◆ Run the DDH oracle on this triplet:
 $\langle g^k, y \cdot g^v, (m \cdot y^k) \cdot g^{kv} / m_0 \rangle = \langle g^k, g^{x+v}, m \cdot g^{(x+v)k} / m_0 \rangle$
 - v is random
- ◆ If this is a DH triplet, then $m=m_0$, else $m=m_1$
- ◆ This breaks semantic security of ElGamal (why?)

(1) ElGamal \Rightarrow DDH

- ◆ Suppose some algorithm A breaks ElGamal
 - Given any public key, A produces plaintexts m_0 and m_1 whose encryptions it can distinguish with advantage Adv
- ◆ We will use A to break DDH
 - Decide, given (g^a, g^b, Z) , whether $Z = g^{ab} \pmod p$ or not
- ◆ Give $y = g^a \pmod p$ to A as the public key
- ◆ A produces m_0 and m_1
- ◆ Toss a coin for bit x and give A the ciphertext $(g^b, m_x \cdot Z) \pmod p$
 - This is a valid ElGamal encryption of m_x **iff** $Z = g^{ab} \pmod p$

(2) ElGamal \Rightarrow DDH

- ◆ A receives $(g^b, m_x \cdot Z) \pmod p$
 - This is a valid ElGamal encryption of m_x **iff** $Z = g^{ab} \pmod p$
- ◆ A outputs his guess of bit x (**why?**)
- ◆ If A guessed x correctly, we say that $Z = g^{ab} \pmod p$, otherwise we say that Z is random
- ◆ What is our advantage in breaking DDH?
 - If $Z = g^{ab} \pmod p$, we are correct with probability $\text{Adv}(A)$
 - If Z is random, we are correct with probability $1/2$
 - Our advantage in breaking DDH is $\text{Adv}(A)/2$

Beyond Semantic Security

◆ Chosen-ciphertext security

- “Lunch-time” attack [Naor and Yung 1990]
- Adaptive chosen-ciphertext security [Rackoff and Simon 1991]

◆ Non-malleability [Dolev, Dwork, Naor 1991]

- Infeasible to create a “related” ciphertext
- Implies that an encrypted message cannot be modified without decrypting it
- Equivalent to adaptive chosen-ciphertext security