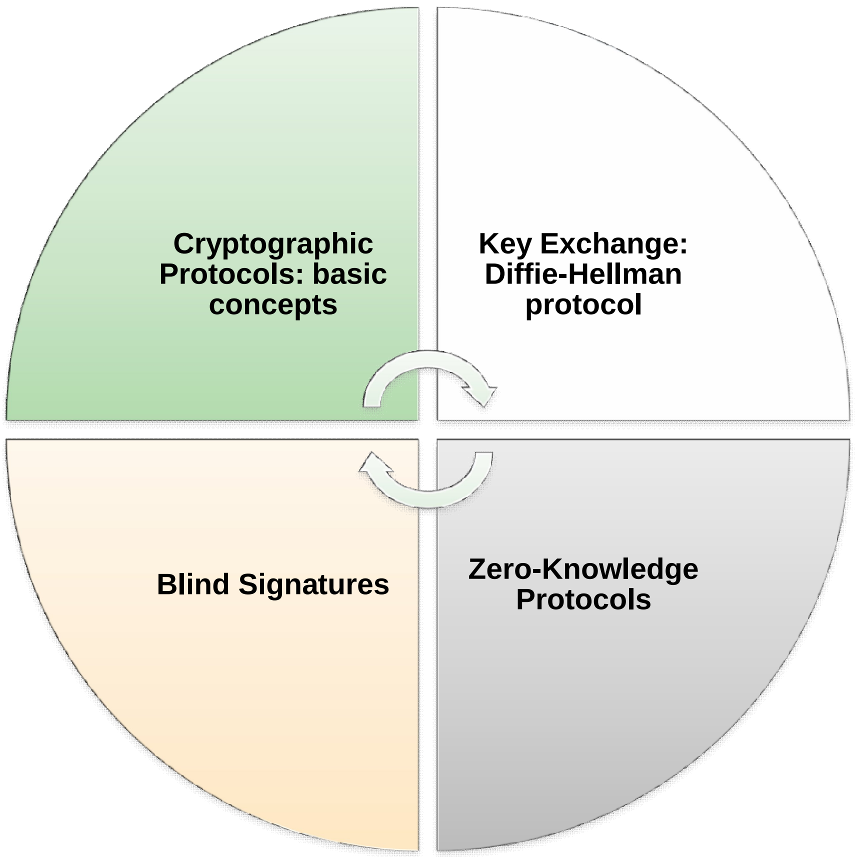
Cryptographic Protocols II

Van Nguyen – HUT
Hanoi – 2010

Agenda


Cryptographic Protocols: basic concepts
Key Exchange: Diffie-Hellman protocol
Blind Signatures
Zero-Knowledge Protocols

Whittfield Diffie and Martin Hellman are called the inventors of Public Key Cryptography. Diffie-Hellman Key Exchange is the first Public Key Algorithm published in 1976.

DIFFIE-HELLMAN KEY EXCHANGE

What is Diffie-Hellman?

- A Public Key Algorithm
- Only for Key Exchange
- Does NOT Encrypt or Decrypt
- Based on Discrete Logarithms
- Widely used in Security Protocols and Commercial Products
- Williamson of Britain's CESG claims to have discovered it several years prior to 1976

Discrete Logarithms

- What is a logarithm?
- $log_{10}100 = 2$ because $10^2 = 100$
- In general if $log_m b = a$ then $m^a = b$
- Where m is called the **base** of the logarithm
- A **discrete logarithm** can be defined for integers only
- In fact we can define **discrete logarithms** *mod p* also where p is any prime number

Discrete Logarithm Problem

- The security of the Diffie-Hellman algorithm depends on the difficulty of solving the discrete logarithm problem (DLP) in the multiplicative group of a finite field

Sets, Groups and Fields

- A set is any collection of objects called the elements of the set
- Examples of sets: ***R***, ***Z***, ***Q***
- If we can define an operation on the elements of the set and certain rules are followed then we get other mathematical structures called groups and fields

Groups

- A group is a set G with a custom-defined binary operation + such that:
  - The group is closed under +, i.e., for a, $b \in G$:
    - $a + b \in G$
  - The Associative Law holds i.e., for any a, b, $c \in G$:
    - $a + (b + c) = (a + b) + c$
  - There exists an identity element 0, such that
    - $a + 0 = a$
  - For each $a \in G$ there exists an inverse element $-a$ such that
    - $a + (-a) = 0$
- If for all a, $b \in G$: $a + b = b + a$ then the group is called an Abelian or commutative group
- If a group G has a finite number of elements it is called a finite group

More About Group Operations

- + does not necessarily mean normal arithmetic addition
- + just indicates a binary operation which can be custom defined
- The group operation could be denoted as •
- The group notation with + is called the additive notation and the group notation with • is called the multiplicative notation

Fields

- A field is a set F with two custom-defined binary operations + and • such that:
  - The Field is closed under + and •, i.e., for a, $b \in F$:
    - $a + b \in F$ and $a \bullet b \in F$
  - The Associative Law holds i.e., for any a, b, $c \in F$:
    - $a + (b + c) = (a + b) + c$ and $a \bullet (b \bullet c) = (a \bullet b) \bullet c$
  - There exist identity elements 0 and 1, such that
    - $a + 0 = a$ and $a \bullet 1 = a$
  - For each $a \in F$ there exist inverse elements $-a$ and a^{-1} such that
    - $a + (-a) = 0$ and $a \bullet a^{-1} = 1$
- If a field F has a finite number of elements it is called a finite field

Examples of Groups

- Groups
  - Set of real numbers ***R*** under +
  - Set of real numbers ***R*** under *
  - Set of integers ***Z*** under +
  - Set of integers ***Z*** under *?
  - Set of integers modulo a prime number *p* under +
  - Set of integers modulo a prime number *p* under *
  - Set of 3 X 3 matrices under + meaning matrix addition
  - Set of 3 X 3 matrices under * meaning matrix multiplication?
- Fields
  - Set of real numbers ***R*** under + and *
  - Set of integers ***Z*** under + and *
  - Set of integers modulo a prime number *p* under + and *

Generator of Group

- If for $a \in G$, all members of the group can be written in terms of a by applying the group operation * on a a number of times then a is called a generator of the group G
- Examples
  - 2 is a generator of Z^*_{11}
  - 2 and 3 are generator of Z^*_{19}

m	1	2	3	4	5	6	7	8	9	10
2^m mod 11	2	4	8	5	10	9	7	3	6	1

m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2^m mod 19	2	4	8	16	13	7	14	3	6	17	15	11	3	6	12	5	10	1
3^m mod 19	3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1

Primitive Roots

- If $a^n = x$ then *a* is called the *n-th* root of *x*
- For any prime number *p*, if we have a number *a* such that powers of *a* mod *p* generate all the numbers between *1* to *p-1* then *a* is called a **Primitive Root** of *p*.
- In terms of the Group terminology *a* is the generator element of the multiplicative group of the finite field formed by mod *p*
- Then for any integer *b* and a primitive root *a* of prime number *p* we can find a unique exponent *i* such that

$$b = a^i \bmod p$$

- The exponent *i* is referred to as the **discrete logarithm** or **index**, of *b* for the base *a*.

Table 7.6 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Table 7.7 Tables of Discrete Logarithms, Modulo 19

(a) Discrete logarithms to the base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{Ind}_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) Discrete logarithms to the base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{Ind}_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Discrete logarithms to the base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{Ind}_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Discrete logarithms to the base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{Ind}_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Discrete logarithms to the base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{Ind}_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	14	9

(f) Discrete logarithms to the base 15, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{Ind}_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	12	9

Diffie-Hellman Algorithm

- Five Parts
  1. Global Public Elements
  2. User A Key Generation
  3. User B Key Generation
  4. Generation of Secret Key by User A
  5. Generation of Secret Key by User B

Global Public Elements

- q Prime number
- α $\alpha < q$ and α is a primitive root of q
- The global public elements are also sometimes called the domain parameters

User A Key Generation

- Select private X_A $X_A < q$
- Calculate public Y_A $Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

- Select private X_B $\quad X_B < q$
- Calculate public Y_B $\quad Y_B = \alpha^{X_B} \bmod q$

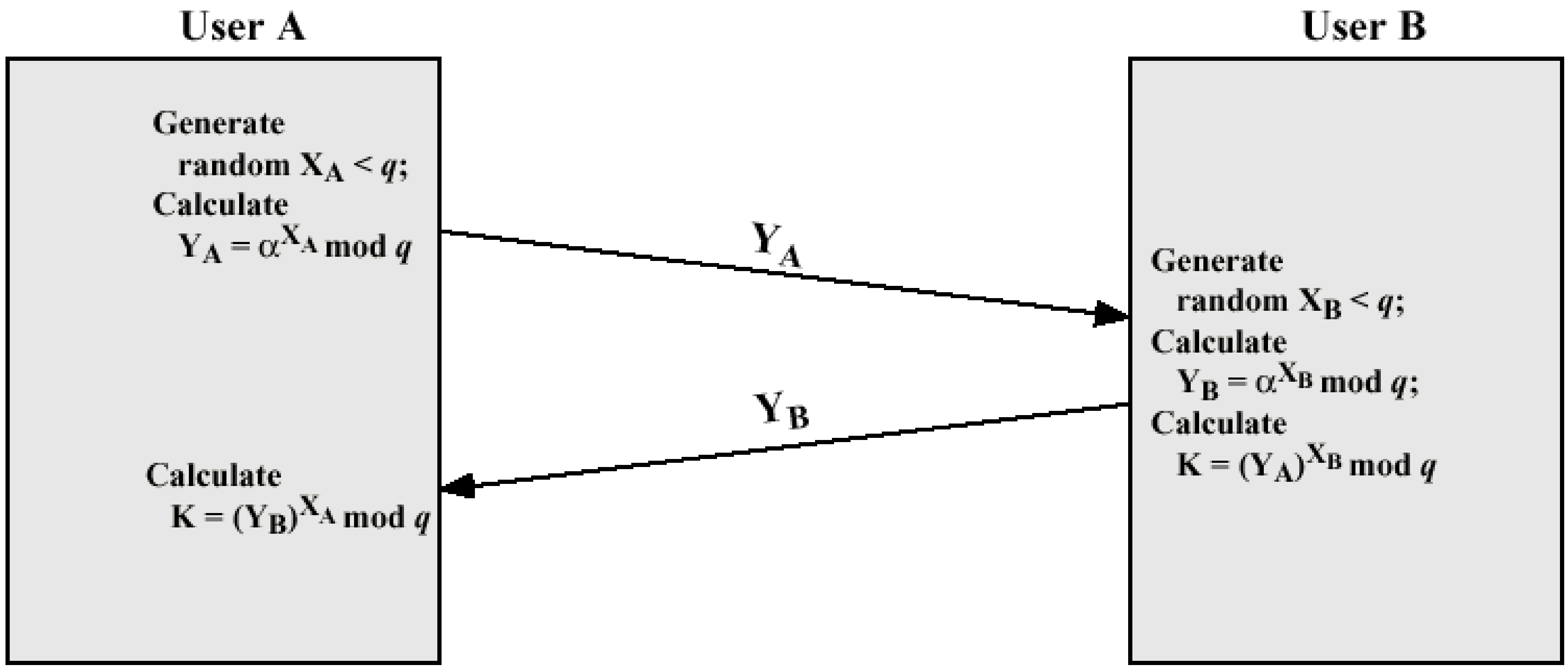
Generation of Secret Key by User A

- $K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

- $K = (Y_A)^{X_B} \bmod q$

Diffie-Hellman Key Exchange



Diffie-Hellman Example

- $q = 97$
- $\alpha = 5$
- $X_A = 36$
- $X_B = 58$
- $Y_A = 5^{36} = 50 \bmod 97$
- $Y_B = 5^{58} = 44 \bmod 97$
- $K = (Y_B)^{X_A} \bmod q = 44^{36} \bmod 97 = 75 \bmod 97$
- $K = (Y_A)^{X_B} \bmod q = 50^{58} \bmod 97 = 75 \bmod 97$

Why Diffie-Hellman is Secure?

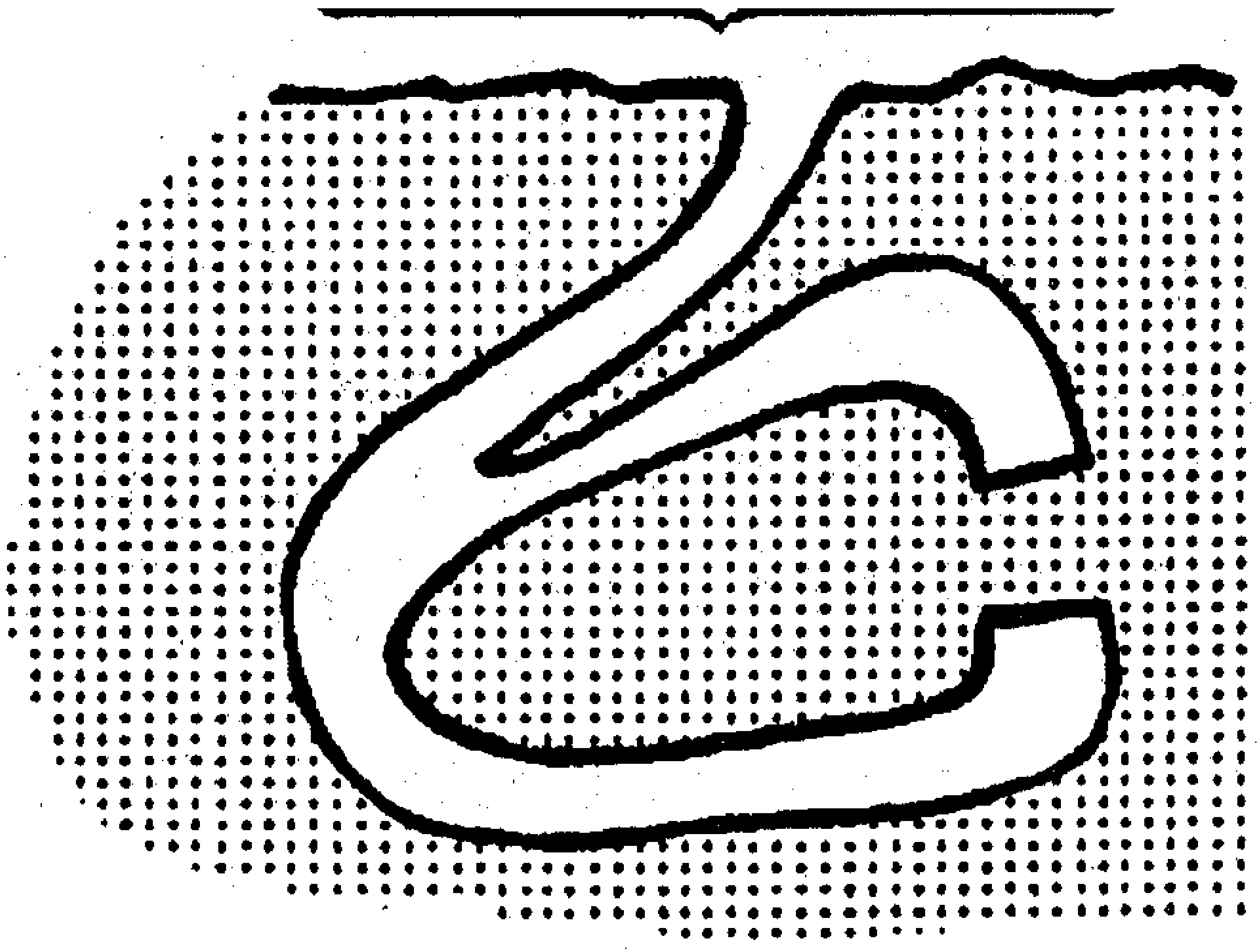
- Opponent has q, α, Y_A and Y_B
- To get X_A or X_B the opponent is forced to take a discrete logarithm
- The security of the Diffie-Hellman Key Exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

Homework: Man-in-the-middle-attack

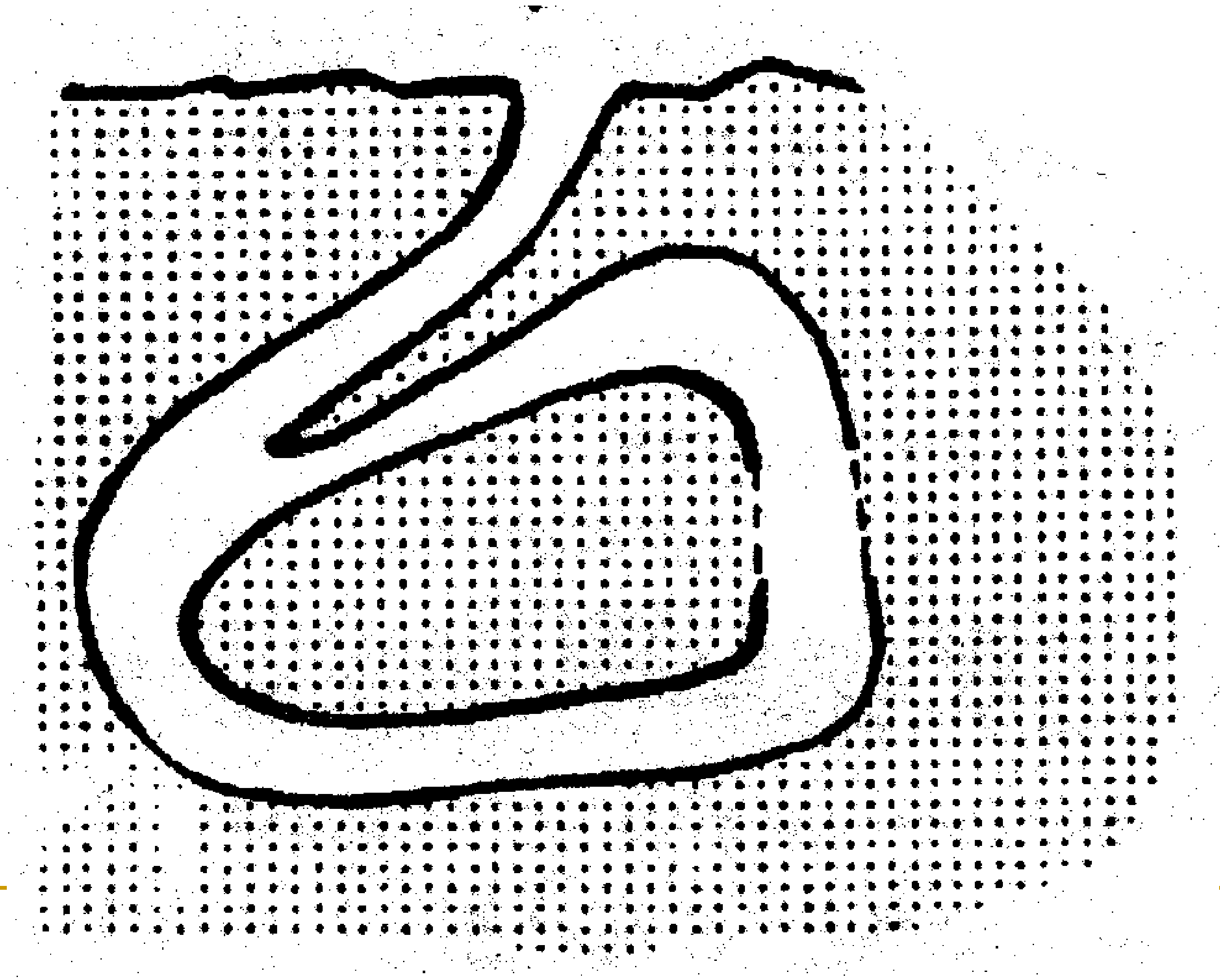
- Find out yourself (maybe from the Internet) how this attack can be implemented successfully against Diffie-Hellman protocol
- Suggest a way to prevent against this attack

ZERO-KNOWLEDGE PROTOCOLS

The Cave of the Forty Thieves



The Cave of the Forty Thieves



Properties of Zero-Knowledge Proofs

- Completeness – A prover who knows the secret information can prove it with probability 1.
- Soundness – The probability that a prover who does not know the secret information can get away with it can be made arbitrarily small.

An Example

- Peggy the prover would like to show Vic the verifier that an element β is a member of the subgroup of Z_n^* generated by α, where α has order λ. (i.e., does $\alpha^k = \beta$ for some k such that $0 \leq k \leq \lambda$?)
- Peggy chooses a random j, $0 \leq j \leq \lambda - 1$, and sends Vic α^j.
- Vic chooses a random i = 0 or 1, and sends it to Peggy.
- Peggy computes $j + ik \bmod \lambda$, and sends it to Vic.
- Vic checks that $\alpha^{j + ik} = \alpha^j\alpha^{ik} = \alpha^j\beta^i$.
- They then repeat the above steps $\log_2 n$ times.
- If Vic's final computation checks out in each round, he accepts the proof.

Complexity Theory

- The last proof works because the problem of solving discrete logarithms is NP-complete (or is believed to be, at any rate).
- It has been shown that all problems in NP have a zero-knowledge proof associated with them.

Identification

- Alice is identified by some secret she alone is known to possess - e.g. a password
- Problems
  - The authenticator must be trusted
  - If secret sniffed or given to untrusted party, can impersonate
- Use zero knowledge!

Fiat-Shamir Identification

One time setup:

- Trusted center published modulus $n=pq$, but keeps p and q secret
- Alice selects a secret prime s comprime to n, computes $v=s^2 \mod n$, and registers v with the trusted center as its public key

Fiat-Shamir Identification

Protocol messages:

$A \rightarrow B$: $x = r^2 \bmod n$

$B \rightarrow A$: e from $\{0, 1\}$

$A \rightarrow B$: $y = rs^e \bmod n$

Fiat-Shamir Identification

Protocol messages:

$A \rightarrow B$: $x = r^2$

$B \rightarrow A$: e from $\{0, 1\}$

$A \rightarrow B$: $y = rs^e \bmod n$

If e=0, then the response y=r is independent of secret s

Fiat-Shamir Identification

Protocol messages:

$A \rightarrow B$: $x = r^2 \bmod n$

$B \rightarrow A$: e from $\{0, 1\}$

$A \rightarrow B$: $y = rs^e \bmod n$

If e=1, then information pairs (x, y) can be simulated by choosing y randomly, and setting $x=y^2 /v \bmod n$

Bit Commitments

- “Flipping a coin down a well”
- “Flipping a coin by telephone”
- A value of 0 or 1 is committed to by the prover by encrypting it with a one-way function, creating a “blob”. The verifier can then “unwrap” this blob when it becomes necessary by revealing the key.

Bit Commitment Properties

- Concealing – The verifier cannot determine the value of the bit from the blob.
- Binding – The prover cannot open the blob as both a zero and a one.

Bit Commitments: An Example

- Let $n = pq$, where p and q are prime. Let m be a quadratic nonresidue modulo n. The values m and n are public, and the values p and q are known only to Peggy.
- Peggy commits to the bit b by choosing a random x and sending Vic the blob $m^b x^2$.
- When the time comes for Vic to check the value of the bit, Peggy simply reveals the values b and x.
- Since no known polynomial-time algorithm exists for solving the quadratic residues problem modulo a composite n whose factors are unknown, hence this scheme is computationally concealing.
- On the other hand, it is perfectly binding, since if it wasn't, m would have to be a quadratic residue, a contradiction.

Bit Commitments and Zero-Knowledge

- Bit commitments are used in zero-knowledge proofs to encode the secret information.
- For example, zero-knowledge proofs based on graph colorations exist. In this case, bit commitment schemes are used to encode the colors.
- Complex zero-knowledge proofs with large numbers of intermediate steps that must be verified also use bit commitment schemes.

- Consider this simple protocol
  1. If the prover claims to be A, the verifier chooses a random message M, and sends the ciphertext $C = P_A(M)$ to the prover.
  2. The prover decrypts C using S_A and sends the result M' to the verifier.
  3. The verifier accepts the identity of the prover if and only if M' = M.
- At first sight, it may seem OK:
  - V already knows M
- But WRONG! What if the verifier is an adversary?

Fixed protocol

1. If P claims to be A, V chooses a random message M, and sends C = P_A (M) to P
2. P decrypts C using S_A and sends V a commitment to the result $commit_{pk}$(r,M')
3. V ➔ P: M.
4. P checks if M = M'. If not he stops the protocol. Otherwise he opens the commitment P➔V: r,M
5. V accepts the identity of the prover if and only if M' = M and the pair r,M' correctly opens the commitment

Computational Assumptions

- A zero-knowledge proof assumes the prover possesses unlimited computational power.
- It is more practical in some cases to assume that the prover's computational abilities are bounded. In this case, we have a zero-knowledge argument.

Proof vs. Argument

Zero-Knowledge Proof:

- Unconditional completeness
- Unconditional soundness
- Computational zero-knowledge
- Unconditionally binding blobs
- Computationally concealing blobs

Zero-Knowledge Argument:

- Unconditional completeness
- Computational soundness
- Perfect zero-knowledge
- Computationally binding blobs
- Unconditionally concealing blobs

Applications

- Zero-knowledge proofs can be applied where secret knowledge too sensitive to reveal needs to be verified
- Key authentication
- PIN numbers
- Smart cards

Limitations

- A zero-knowledge proof is only as good as the secret it is trying to conceal
- Zero-knowledge proofs of identities in particular are problematic
- The Grandmaster Problem
- The Mafia Problem
- etc.