KRYPTOLOGIE II

Ausgewählte Folien zur Vorlesung

Wintersemester 2009/2010

Dozent: Prof. Dr. J. Rothe

Heinrich-Heine-Universität Düsseldorf

`http://ccc.cs.uni-duesseldorf.de/~rothe/crypto2`

Literatur

- **Jörg Rothe: „Komplexitätstheorie und Kryptologie. Eine Einführung in Kryptokomplexität“**, eXamen.press, Springer-Verlag, 2008

- **Jörg Rothe: „Complexity Theory and Cryptology. An Introduction to Cryptocomplexity“**, Springer-Verlag, 2005

- **Douglas R. Stinson: „Cryptography: Theory and Practice“**, Chapman & Hall/CRC, 2. Auflage, 2002

- **Johannes Buchmann: „Einführung in die Kryptographie“**, Springer-Verlag, 2. Auflage, 2001

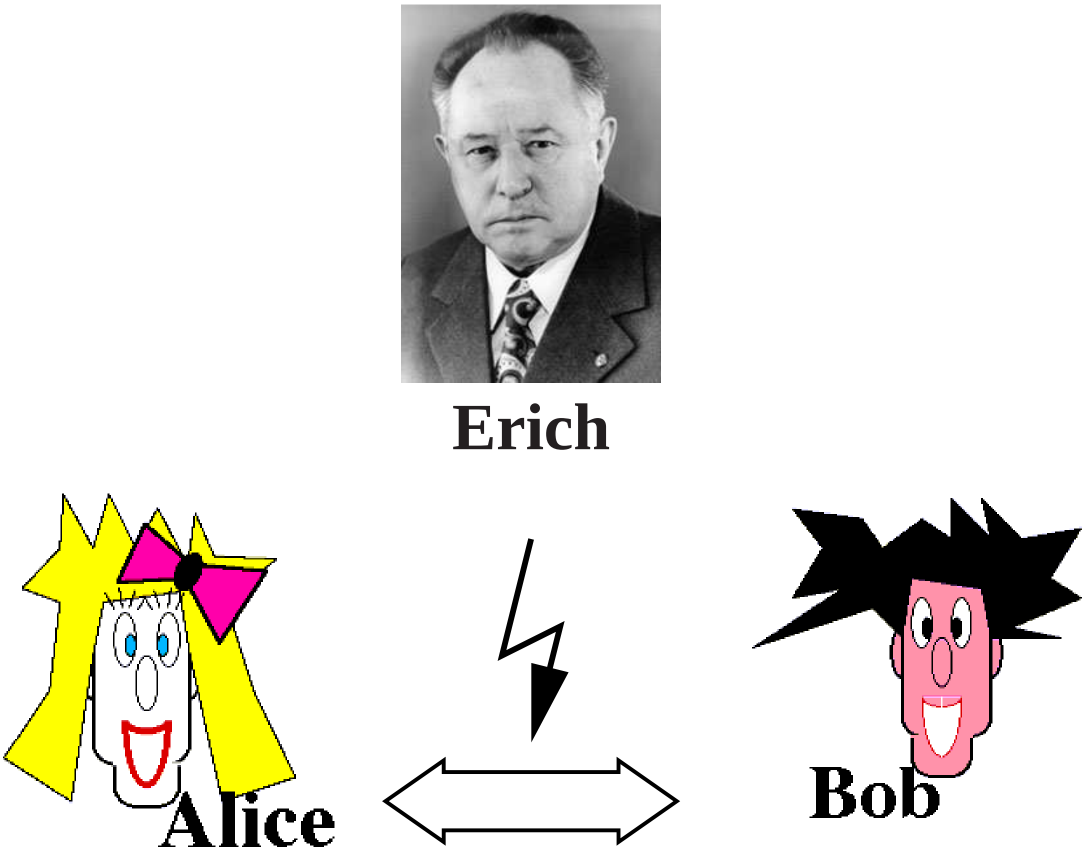
- **Oded Goldreich: „Foundations of Cryptography“**, Cambridge University Press, 2001

- **Neal Koblitz: „Algebraic Aspects of Cryptography“**, Springer-Verlag, 2. Auflage, 1999

- **Bruce Schneier: „Applied Cryptography“**, John Wiley & Sons, 1996

- **Arto Salomaa: „Public-Key Cryptography“**, Springer-Verlag, 1990

Secret-Key Agreement



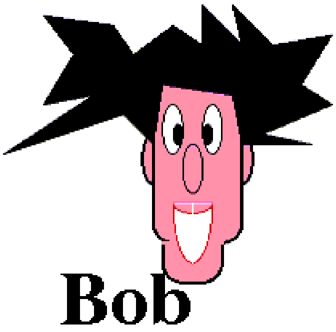
Alice and **Bob** want to agree on a joint secret key k, by communicating over an insecure channel that is eavesdropped by **Erich**.

Method Square-and-multiply

```
SQUARE-AND-MULTIPLY$(a, b, m)$ {
                                  // exponent $a$, base $b < m$, and modulus $m$
    Determine the binary expansion of the exponent
        $a = \sum_{i=0}^{k} a_i 2^i, \qquad \text{where } a_i \in \{0, 1\};$
    Successively, compute
        $b^{2^0}, b^{2^1}, \ldots, b^{2^k}$
      by applying the congruence
        $b^{2^{i+1}} \equiv \left(b^{2^i}\right)^2 \bmod m;$
                                  // the intermediate values $b^{2^i}$ need not be stored
    In the arithmetics modulo $m$, compute
        $b^a = \prod_{\substack{i = 0 \\ a_i = 1}}^{k} b^{2^i};$
    return $b^a$;
}
```

Secret-Key Agreement Protocol

Diffie and Hellman; 1976

		
Alice and Bob agree upon a prime p and a primitive root γ of p; p and γ are public		
chooses a at random, computes $\alpha = \gamma^a \bmod p$		chooses b at random, computes $\beta = \gamma^b \bmod p$
	$\overset{\alpha}{\Rightarrow}$ $\overset{\beta}{\Leftarrow}$	
computes her key $k_A = \beta^a \bmod p$		computes his key $k_B = \alpha^b \bmod p$

Shanks' Baby-Step Giant-Step Algorithm

```
SHANKS$(G, n, \gamma, \alpha)$ {
                    // $G$ is a multiplicative group, $\gamma \in G$ is a
                    // primitive element of order $n$, and $\alpha \in \langle \gamma \rangle$

  $s := \lceil \sqrt{n} \rceil$;

  for $(i = 0, 1, \ldots, s-1)$ { add $(\gamma^{is}, i)$ to a list $\mathcal{L}_1$; }

  Sort the elements of $\mathcal{L}_1$ w.r.t. their first coordinates;

  for $(j = 0, 1, \ldots, s-1)$ { add $(\alpha\gamma^{-j}, j)$ to a list $\mathcal{L}_2$; }

  Sort the elements of $\mathcal{L}_2$ w.r.t their first coordinates;

  Find a pair $(\delta, i) \in \mathcal{L}_1$ and a pair $(\delta, j) \in \mathcal{L}_2$, i.e., find two
     pairs with identical first coordinates;

  return "$\log_\gamma \alpha = is + j$" and halt;
}
```

Example for Shanks' Algorithm

- Let $p = 101$, $\gamma = 2$, and $\alpha = 47$.
- Suppose we want to find

  $$a = \log_2 47 \bmod 101$$

  in the group $\mathbb{Z}_{101}^*$.
- Since $n = p - 1 = 100$ is the order of 2, we have

  $$s = \left\lceil \sqrt{100} \right\rceil = 10.$$

- It follows that

  $$\gamma^s \bmod p = 2^{10} \bmod p = 14.$$

- Now, the sorted lists $\mathcal{L}_1$ and $\mathcal{L}_2$ can be determined as follows:

$\mathcal{L}_1$	$(1,0)$	$(14,1)$	$(95,2)$	$(17,3)$	$(36,4)$	$(100,5)$	$(87,6)$	$(6,7)$	$(84,8)$	$(65,9)$
$\mathcal{L}_1$ sorted	$(1,0)$	$(6,7)$	$(14,1)$	$(17,3)$	$(36,4)$	$(65,9)$	$(84,8)$	$(87,6)$	$(95,2)$	$(100,5)$

$\mathcal{L}_2$	$(47,0)$	$(74,1)$	$(37,2)$	$(69,3)$	$(85,4)$	$(93,5)$	$(97,6)$	$(99,7)$	$(100,8)$	$(50,9)$
$\mathcal{L}_2$ sorted	$(37,2)$	$(47,0)$	$(50,9)$	$(69,3)$	$(74,1)$	$(85,4)$	$(93,5)$	$(97,6)$	$(99,7)$	$(100,8)$

Pohlig-Hellman’s Algorithm

```
POHLIG-HELLMAN($G, n, \gamma, \alpha, q, c$) {
                              // $G$ is a multiplicative group of order $n$,
                    // $\gamma \in G$ is a primitive element, $\alpha \in \langle\gamma\rangle$, prime $q$,
                                   // $n \equiv 0 \bmod q^c$ and $n \not\equiv 0 \bmod q^{c+1}$

  $j := 0$;
  $\alpha_j := \alpha$;
  while ($j \leq c - 1$) {
    $\delta := \alpha_j^{n/q^{j+1}}$;
    Find $i$ with $\delta = \gamma^{in/q}$;
    $a_j := i$;
    $\alpha_{j+1} := \alpha_j \gamma^{-a_j q^j}$;
    $j := j + 1$;
  }
  return “$(a_0, a_1, \ldots, a_{c-1})$” and halt;
}
```

Der Chinesische Restesatz (CRS)

Satz 1 (Chinese Remainder Theorem)
Let $m_1, m_2, \ldots, m_k$ *be* k *positive integers that are pairwise relatively prime (i.e.,* $\texttt{gcd}(m_i, m_j) = 1$ *for* $i \neq j$*), let*

$$M = \prod_{i=1}^{k} m_i,$$

and let $a_1, a_2, \ldots, a_k$ *be any integers.*
For each i *with* $1 \leq i \leq k$*, define* $q_i = M/m_i$*, and let* q_i^{-1} *denote the inverse element of* q_i *in* $\mathbb{Z}_{m_i}^*$*.*
Then, the system of k *congruences*

$$x \equiv a_i \bmod m_i,$$

where $1 \leq i \leq k$*, has the unique solution*

$$x = \sum_{i=1}^{k} a_i q_i q_i^{-1} \bmod M.$$

ElGamal's Public-Key Cryptosystem

Step	**Alice**	**Erich**	**Bob**
1	Alice and Bob agree upon a large prime p and a primitive element γ of p; p and γ are public		
2			chooses a large random number b as his private key and computes $\beta = \gamma^b \bmod p$
3		$\Leftarrow \beta$	
4	chooses a large random number a and encrypts message m by: $\alpha_1 = \gamma^a \bmod p$ $\alpha_2 = m\beta^a \bmod p$		
5		$(\alpha_1, \alpha_2) \Rightarrow$	
6			decrypts by $\alpha_2 \, (\alpha_1)^{-b} \bmod p$

ElGamal’s Digital Signature Scheme

Step	**Alice**	**Erich**	**Bob**
1			chooses a large prime p, a primitive element γ of p, and a large private number b and computes $\beta = \gamma^b \bmod p$
2		$\Leftarrow (p, \gamma, \beta)$	
3			chooses a large random s with $\gcd(s, p-1) = 1$, and computes his signature for message m by $\mathrm{sig}_B(m) = (\sigma, \rho)$, where $\sigma = \gamma^s \bmod p$ $\rho = (m - b\sigma)s^{-1} \bmod p - 1$
4		$\Leftarrow \begin{cases} m \\ (\sigma, \rho) \end{cases}$	
5	verifies Bob’s signature by checking $\gamma^m \equiv \beta^\sigma \sigma^\rho \bmod p$		

ElGamal's Digital Signature Scheme

Verifying Bob's Signature

- Let $p = 1367$, and let $\gamma = 5$ be a primitive element of 1367.
- Bob chooses the private exponents $b = 513$ and $s = 129$, where $\texttt{gcd}(129, 1366) = 1$.
- Bob computes

$$\begin{aligned} \beta &= 5^{513} \bmod 1367 = 855 \quad \text{and} \\ \sigma &= 5^{129} \bmod 1367 = 1180. \end{aligned}$$

- Suppose that Bob wants to sign the message $m = 457$.

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	
5	25	625	1030	108	728	955	236	1016			$\gamma^m \bmod p$
855	1047	1242	588	1260	513	705	804	1192	551	127	$\beta^\sigma \bmod p$
1180	794	249	486	1072	904	1117	985	1022	96		$\sigma^\rho \bmod p$

Thus, $\gamma^m = 1280$, $\beta^\sigma = 749$, and $\sigma^\rho = 750$.

Gray Boxes contain the values to be multiplied according to the binary expansion of the exponents:

$$\begin{aligned} m &= 457 = 2^0 + 2^3 + 2^6 + 2^7 + 2^8; \\ \sigma &= 1180 = 2^2 + 2^3 + 2^4 + 2^7 + 2^{10}; \\ \rho &= 955 = 2^0 + 2^1 + 2^3 + 2^4 + 2^5 + 2^7 + 2^8 + 2^9. \end{aligned}$$

Discrete Logarithm Bit Problem

- Consider $\log_2 47 \bmod 101 = 58$.
- Since
$$58 = 2^5 + 2^4 + 2^3 + 2^1,$$
the binary representation of 58 is
$$\texttt{bin}(58) = 111010$$
and has six bits, dropping leading zeros.
- The *least significant bit of* $\texttt{bin}(58)$ is the rightmost zero, the coefficient of 2^0.

i	1	2	3	4	5	6	7
$\texttt{DLogBit}(\langle 101, 2, 47, i\rangle)$	0	1	0	1	1	1	0

Discrete Logarithm Reducing to DLogBit(. . . , 2)

```
DISCRETE-LOG-BIT-2-ORACLE$(p, \gamma, \alpha)$ {
                                        // $p$ is prime,
                                        // $\gamma$ is a primitive root of $p$,
                                        // $\alpha \in \langle\gamma\rangle$
 // external: Algo for DLogBit$(\ldots, 1)$ and DLogBit$(\ldots, 2)$

   $x_0 :=$ DLogBit$(p, \gamma, \alpha, 1)$;
   $\alpha := \alpha/\gamma^{x_0} \bmod p$;
   $i := 1$;
   while $(\alpha \neq 1)$ {
      $x_i :=$ DLogBit$(p, \gamma, \alpha, 2)$;
      $\delta := \alpha^{(p+1)/4} \bmod p$;
      if DLogBit$(p, \gamma, \delta, 1) = x_i$
         then $\alpha := \delta$
         else $\alpha := p - \delta$;
      $\alpha := \alpha/\gamma^{x_i} \bmod p$;
      $i := i + 1$;
   }
   return "$(x_{i-1}, x_{i-2}, \ldots, x_0)$";
}
```

Example for DLog Reducing to DLogBit(. . . , 2)

Let $p = 19$, $\gamma = 2$, and $\alpha = 6$.

We want to compute $\log_\gamma \alpha \bmod p = \log_2 6 \bmod 19$.

The following table gives the values of $\texttt{DLogBit}(19, 2, \beta, 1)$ and $\texttt{DLogBit}(19, 2, \beta, 2)$ for each $\beta \in \mathbb{Z}^*_{19}$:

β	$\texttt{DLogBit}(19, 2, \beta, 1)$	$\texttt{DLogBit}(19, 2, \beta, 2)$
1	0	0
2	1	0
3	1	0
4	0	1
5	0	0
6	0	1
7	0	1
8	1	1
9	0	0
10	1	0
11	0	0
12	1	1
13	1	0
14	1	1
15	1	1
16	0	0
17	0	1
18	1	0

Types of Forgery

- **Total Break:** The cryptanalyst is able to determine the private key of the sender in a digital signature scheme.

  For example, Bob's secret number b in ElGamal's digital signature scheme.

  Using this private key, cryptanalyst Erich can create a valid signature for any message of his choice.

- **Selective Forgery:** The cryptanalyst is able to create, with nonnegligible probability of success, a valid signature for some message chosen by somebody else.

  If Erich intercepts a message m that was previously not signed by Bob, he is able to create a valid signature for m with a certain success probability.

- **Existential Forgery:** The cryptanalyst is able to create a valid signature for at least one message that was previously not signed by Bob.

  Here, no specified probability of success is required.

Types of Attacks

- **Key-Only Attack:**
  Cryptanalyst Erich only knows Bob’s public key.

- **Known-Message Attack:**
  Erich knows some pairs of messages and corresponding signatures in addition to the public key.

- **Chosen-Message attack:**
  Erich knows the public key and obtains a list of Bob’s signatures corresponding to a list of messages he has chosen at will.

Security of ElGamal Signatures

Key-Only Attack – Existential Forgery

- Let x and y be integers with
$$0 \leq x \leq p - 2 \quad \text{and} \quad 0 \leq y \leq p - 2.$$

- Writing σ as $\sigma = \gamma^x \beta^y \bmod p$ implies that the ElGamal verification condition
$$\gamma^m \equiv \beta^\sigma \sigma^\rho \bmod p. \tag{1}$$
is of the form
$$\gamma^m \equiv \beta^\sigma \left(\gamma^x \beta^y\right)^\rho \bmod p,$$
which is equivalent to
$$\gamma^{m - x\rho} \ \equiv \ \beta^{\sigma + y\rho} \bmod p. \tag{2}$$

- Equation (2) is true if and only if the following two congruences are satisfied:
$$\begin{aligned} m - x\rho \ &\equiv \ 0 \bmod (p-1); && (3) \\ \sigma + y\rho \ &\equiv \ 0 \bmod (p-1). && (4) \end{aligned}$$

Security of ElGamal Signatures
Key-Only Attack – Existential Forgery
continued

- Given x and y and assuming that $\texttt{gcd}(y, p-1) = 1$, the congruences (3) and (4) can easily be solved for ρ and m, and we obtain:

$$\begin{aligned} \sigma &= \gamma^x \beta^y \bmod p; \\ \rho &= -\sigma y^{-1} \bmod (p-1); \\ m &= -x \sigma y^{-1} \bmod (p-1). \end{aligned}$$

- By way of construction, (σ, ρ) is a valid signature for the message m.

Security of ElGamal Signatures
Known-Message Attack – Existential Forgery

- Suppose that Erich knows a previous signature $(\hat{\sigma}, \hat{\rho})$ for some message $\hat{m}$.
- He can then sign new messages forging Bob's signature.
- Let p be a prime number with primitive element γ, and let β be Bob's public key.
- Let $x, y, z \in \mathbb{Z}_{p-1}$ be chosen such that

$$\texttt{gcd}(x\hat{\sigma} - z\hat{\rho}, p-1) = 1.$$

- Erich computes:

$$\begin{aligned} \sigma &= \hat{\sigma}^x \gamma^y \beta^z \bmod p; \\ \rho &= \hat{\rho}\sigma(x\hat{\sigma} - z\hat{\rho})^{-1} \bmod (p-1); \\ m &= \sigma(x\hat{m} + y\hat{\rho})(x\hat{\sigma} - z\hat{\rho})^{-1} \bmod (p-1). \end{aligned}$$

- One can check that the ElGamal verification condition (1) is satisfied:

$$\gamma^m \equiv \beta^\sigma \sigma^\rho \bmod p.$$

Security of ElGamal Signatures
Known-Message Attack – Total Break

- Bob's secret exponent s must never be revealed!
- If Erich knows s, then it is a matter of routine for him to compute, using

  $$b\sigma + s\rho \equiv m \bmod p-1,$$

  Bob's secret exponent b from m and the signature (σ, ρ) by

  $$b \equiv (m - s\rho)\sigma^{-1} \bmod p-1.$$
- This known-message attack results in a total break of the ElGamal digital signature scheme, and Erich can henceforth forge Bob's signature at will.
- In particular, if the same s is used twice for signing distinct messages, m_1 and m_2, we have
  - a signature (σ, ρ_1) for m_1 and
  - a signature (σ, ρ_2) for m_2.
- Writing $\sigma = \gamma^s$, we have

  $$\beta^\sigma \sigma^{\rho_1} \equiv \gamma^{m_1} \bmod p \quad \text{and} \quad \beta^\sigma \sigma^{\rho_2} \equiv \gamma^{m_2} \bmod p$$

  from which the unknown value s can be determined.

Rabin’s Public-Key Cryptosystem

Step	Alice	Erich	Bob
1			chooses two large random primes, p and q with $$p \equiv q \equiv 3 \bmod 4,$$ keeps them secret, and computes his public key $$n = pq$$
2		$\Leftarrow n$	
3	encrypts the message m by $c = m^2 \bmod n$		
4		$c \Rightarrow$	
5			decrypts c by computing $$m = \sqrt{c} \bmod n$$

A ZPP Computation

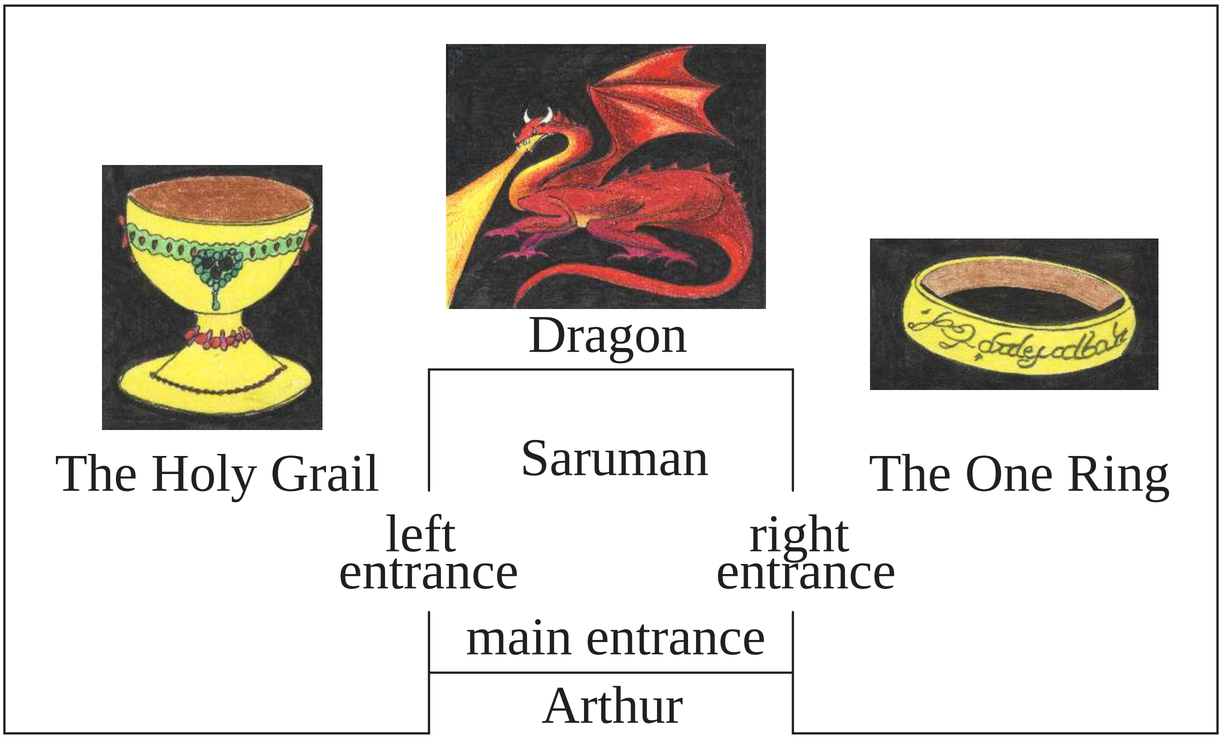
- Let A be any language in ZPP, and let M and N be NPTMs witnessing that $A \in$ RP and $\overline{A} \in$ RP.
- Define the machine $M \circ N$ as follows:
  - On input x, $M \circ N$ first simulates $M(x)$ and then it simulates $N(x)$.
  - Thus, every path of $(M \circ N)(x)$ has the form (α, β), where α is a path of $M(x)$ and β is a path of $N(x)$.
  - For paths α and β, denote acceptance by $+$ and rejection by $-$.
  - $M \circ N$ assigns the final states s_a, s_r, and $s_?$ to each possible pair (α, β) as follows:

	α of $M(x)$	β of $N(x)$	(α, β) of $(M \circ N)(x)$
$x \in A$	$+$	$-$	$(+, -) = s_a$
	$-$	$-$	$(-, -) = s_?$
$x \notin A$	$-$	$+$	$(-, +) = s_r$
	$-$	$-$	$(-, -) = s_?$

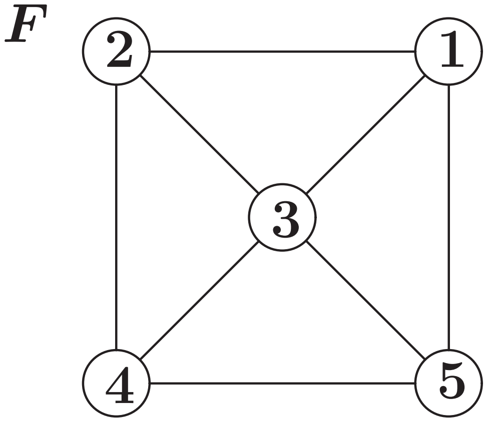
Security of Rabin's System

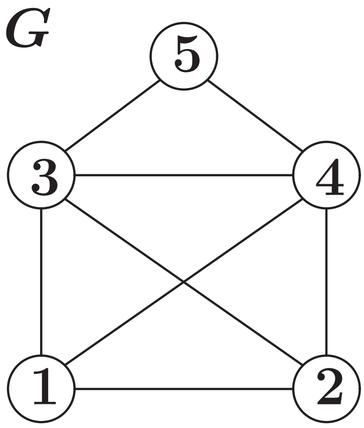
```
RANDOM-FACTOR^{break-rabin}$(n)$ {
            // Rabin modulus $n = pq$ with $p \equiv q \equiv 3 \bmod 4$
  Randomly choose a number $x \in \mathbb{Z}_n^*$ under the uniform
    distribution;
  $c := x^2 \bmod n$;
  $m :=$ break-rabin$(\langle n, c\rangle)$;
                                   // query the oracle about $\langle n, c\rangle$ to
                                   // obtain an $m$ with $c := m^2 \bmod n$
  if $(m \equiv \pm x \bmod n)$ return "failure" and halt;
    else
      $p :=$ gcd$(m - x, n)$;
      $q := n/p$;
      return "$p$ and $q$ are the prime factors of $n$" and halt;
}
```

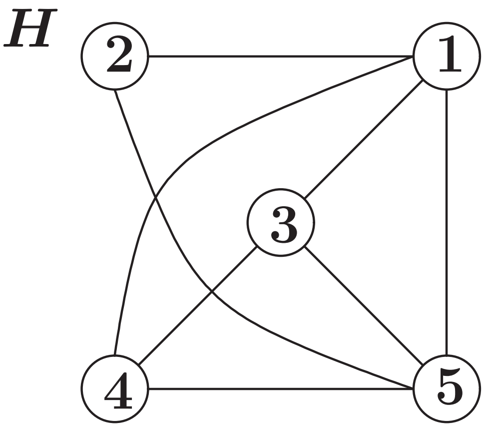
How to Explain Zero-Knowledge to Your Children



The Graph Isomorphism Problem







- G is isomorphic to H, but not to F.
- An isomorphism π between G and H is given by $\pi = \binom{1\,2\,3\,4\,5}{3\,4\,1\,5\,2}$ or, in cyclic notation, by $\pi = (1\,3)(2\,4\,5)$.
- The graph isomorphism problem is neither known to be polynomial-time solvable nor to be NP-complete.

Zero-Knowledge Protocol for Graph Isomorphism

Goldreich, Micali, and Wigderson (J.ACM, 1991)

Step	**Merlin**	**Saruman**	**Arthur**
1	chooses a large graph G_0 with n vertices and a permutation $\pi \in \mathfrak{S}_n$ at random, computes $G_1 = \pi(G_0)$; (G_0, G_1) is public, π secret		
2		$(G_0, G_1) \Rightarrow$	
3	picks a permutation μ in $\mathfrak{S}_n$ and a bit m in $\{0, 1\}$ at random, computes $H = \mu(G_m)$		
4		$H \Rightarrow$	
5			chooses a bit $a \in \{0, 1\}$, requests α in ISO(G_a, H)
6		$\Leftarrow a$	
7	computes $\alpha = \mu$ if $a = m$; $\alpha = \pi\mu$ if $0 = a \neq m = 1$; $\alpha = \pi^{-1}\mu$ if $1 = a \neq m = 0$		
8		$\alpha \Rightarrow$	
9			verifies $\alpha(G_a) = H$

Simulation of the Goldreich–Micali–Wigderson Zero-Knowledge Protocol for Graph Isomorphism

Step	**Saruman**		**Arthur**
1 & 2	Merlin's pair (G_0, G_1) of isomorphic graphs is public information		
3	picks a permutation σ in $\mathfrak{S}_n$ and a bit s in $\{0, 1\}$ at random, computes $H = \sigma(G_s)$		
4		$H \Rightarrow$	
5			chooses a bit $a \in \{0, 1\}$, requests α in ISO(G_a, H)
6		$\Leftarrow a$	
7	sends $\alpha = \sigma$ if $a = s$; deletes this round if $a \neq s$		
8		$\alpha \Rightarrow$	
9			$a = s$ implies $\alpha(G_a) = H$, thus Arthur accepts S's false identity

Fiat–Shamir Zero-Knowledge Identification Scheme

Step	**Merlin**	**Saruman**	**Arthur**
1	chooses two large primes p and q and a secret $s \in \mathbb{Z}_n^*$, and computes $n = pq$ and $v = s^2 \bmod n$		
2		$(n, v) \Rightarrow$	
3	chooses $r \in \mathbb{Z}_n^*$ at random, computes $$x = r^2 \bmod n$$		
4		$x \Rightarrow$	
5			picks a random bit $a \in \{0, 1\}$
6		$\Leftarrow a$	
7	computes $$y = r \cdot s^a \bmod n$$		
8		$y \Rightarrow$	
9			verifies $$y^2 \equiv x \cdot v^a \bmod n$$