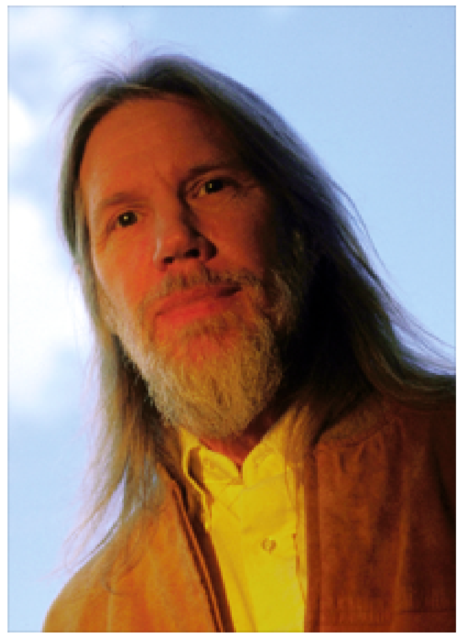
Public-Key Cryptography[a]

- Suppose only d is private to Bob, whereas e is public knowledge.
- Bob generates the (e, d) pair and publishes e.
- Anybody like Alice can send $E(e, x)$ to Bob.
- Knowing d, Bob can recover x by $D(d, E(e, x)) = x$.
- The assumptions are complexity-theoretic.
  - It is computationally difficult to compute d from e.
  - It is computationally difficult to compute x from y without knowing d.

[a] Diffie and Hellman (1976).

Whitfield Diffie (1944–)



Martin Hellman (1945–)

Complexity Issues

- Given y and x, it is easy to verify whether $E(e, x) = y$.
- Hence one can always guess an x and verify.
- Cracking a public-key cryptosystem is thus in NP.
- A necessary condition for the existence of secure public-key cryptosystems is P $\neq$ NP.
- But more is needed than P $\neq$ NP.
- For instance, it is not sufficient that D is hard to compute in the *worst* case.
- It should be hard in "most" or "average" cases.

One-Way Functions

A function f is a **one-way function** if the following hold.[a]

1. f is one-to-one.
2. For all $x \in \Sigma^*$, $|\,x\,|^{1/k} \leq |f(x)| \leq |\,x\,|^k$ for some $k > 0$.
   - f is said to be **honest**.
3. f can be computed in polynomial time.
4. f^{-1} cannot be computed in polynomial time.
   - Exhaustive search works, but it must be slow.

---

[a] Diffie and Hellman (1976); Boppana and Lagarias (1986); Grollmann and Selman (1988); Ko (1985); Ko, Long, and Du (1986); Watanabe (1985); Young (1983).

Existence of One-Way Functions

- Even if $\mathrm{P} \neq \mathrm{NP}$, there is no guarantee that one-way functions exist.
- No functions have been proved to be one-way.
- Is breaking glass a one-way function?

Candidates of One-Way Functions

- Modular exponentiation $f(x) = g^x \bmod p$, where g is a primitive root of p.
  - **Discrete logarithm** is hard.[a]
- The RSA[b] function $f(x) = x^e \bmod pq$ for an odd e relatively prime to $\phi(pq)$.
  - Breaking the RSA function is hard.

[a] Conjectured to be 2^{n^ϵ} for some $\epsilon > 0$ in both the worst-case sense and average sense. Doable in time $n^{O(\log n)}$ for finite fields of small characteristic (Barbulescu, et al., 2013). It is in NP in some sense (Grollmann and Selman (1988)).

[b] Rivest, Shamir, and Adleman (1978).

Candidates of One-Way Functions (concluded)

- Modular squaring $f(x) = x^2 \bmod pq$.
  - Determining if a number with a Jacobi symbol 1 is a quadratic residue is hard—the **quadratic residuacity assumption** (**QRA**).[a]

[a] Due to Gauss.

The RSA Function

- Let p, q be two distinct primes.
- The RSA function is $x^e \bmod pq$ for an odd e relatively prime to $\phi(pq)$.
  - By Lemma 53 (p. 466),

$$\phi(pq) = pq \left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{q}\right) = pq - p - q + 1. \quad (15)$$

- As $\gcd(e, \phi(pq)) = 1$, there is a d such that

$$ed \equiv 1 \bmod \phi(pq),$$

which can be found by the Euclidean algorithm.[a]

[a] One can think of d as e^{-1}.

A Public-Key Cryptosystem Based on RSA

- Bob generates p and q.
- Bob publishes pq and the encryption key e, a number relatively prime to $\phi(pq)$.
  - The encryption function is $y = x^e \bmod pq$.
  - Bob calculates $\phi(pq)$ by Eq. (15) (p. 640).
  - Bob then calculates d such that $ed = 1 + k\phi(pq)$ for some $k \in \mathbb{Z}$.
- The decryption function is $y^d \bmod pq$.
- It works because $y^d = x^{ed} = x^{1+k\phi(pq)} = x \bmod pq$ by the Fermat-Euler theorem when $\gcd(x, pq) = 1$ (p. 477).

The "Security" of the RSA Function

- Factoring pq or calculating d from (e, pq) seems hard.
  - See also p. 473.
- Breaking the last bit of RSA is as hard as breaking the RSA.[a]
- Recommended RSA key sizes:[b]
  - 1024 bits up to 2010.
  - 2048 bits up to 2030.
  - 3072 bits up to 2031 and beyond.

---

[a] Alexi, Chor, Goldreich, and Schnorr (1988).

[b] RSA (2003). RSA was acquired by EMC in 2006 for 2.1 billion US dollars.

The "Security" of the RSA Function (continued)

- Recall that problem A is "harder than" problem B if solving A results in solving B.
  - Factorization is "harder than" breaking the RSA.
  - It is not hard to show that calculating Euler's phi function[a] is "harder than" breaking the RSA.
  - Factorization is "harder than" calculating Euler's phi function (see Lemma 53 on p. 466).
  - So factorization is harder than calculating Euler's phi function, which is harder than breaking the RSA.

---

[a]When the input is not factorized!

The "Security" of the RSA Function (concluded)

- Factorization cannot be NP-hard unless NP = coNP.[a]
- So breaking the RSA is unlikely to imply P = NP.
- RSA was alleged to have received 10 million US dollars from the government to promote an unsecure software![b]

---

[a]Brassard (1979).

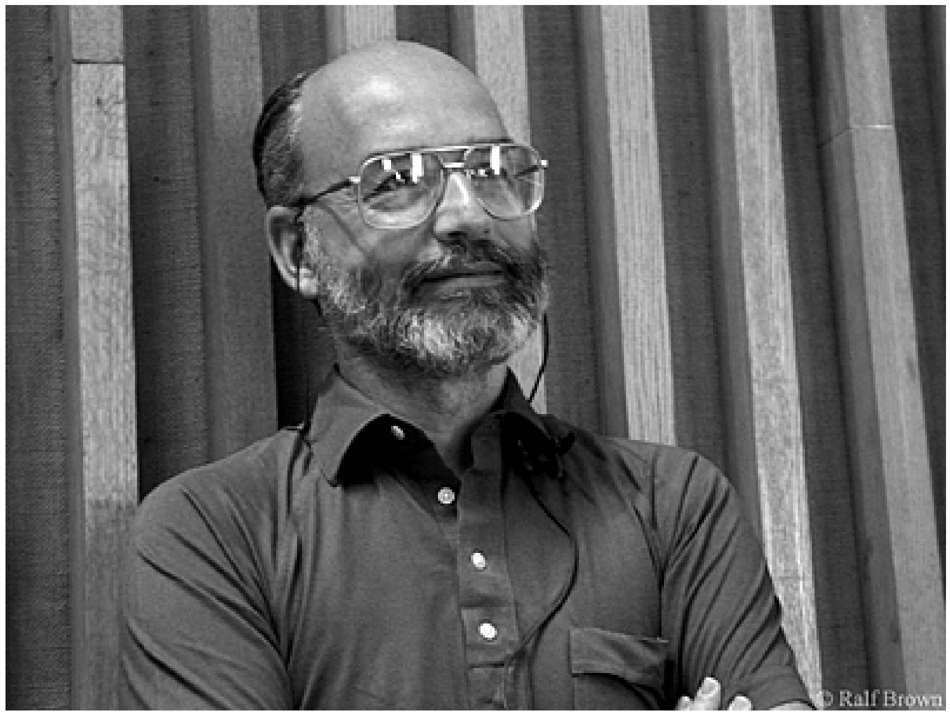
[b]Menn (2013).

Adi Shamir, Ron Rivest, and Leonard Adleman

Ron Rivest[a] (1947–)

[a]Turing Award (2002).

Adi Shamir[a] (1952–)



[a]Turing Award (2002).

The Secret-Key Agreement Problem

- Exchanging messages securely using a private-key cryptosystem requires Alice and Bob possessing the same key (p. 631).
- How can they agree on the same secret key when the channel is insecure?
- This is called the **secret-key agreement problem**.
- It was solved by Diffie and Hellman (1976) using one-way functions.

The Diffie-Hellman Secret-Key Agreement Protocol

1: Alice and Bob agree on a large prime p and a primitive root g of p; $\{p$ and g are public.$\}$

2: Alice chooses a large number a at random;

3: Alice computes $\alpha = g^a \bmod p$;

4: Bob chooses a large number b at random;

5: Bob computes $\beta = g^b \bmod p$;

6: Alice sends α to Bob, and Bob sends β to Alice;

7: Alice computes her key $\beta^a \bmod p$;

8: Bob computes his key $\alpha^b \bmod p$;

Analysis

- The keys computed by Alice and Bob are identical as

$$\beta^a = g^{ba} = g^{ab} = \alpha^b \bmod p.$$

- To compute the common key from p, g, α, β is known as the **Diffie-Hellman problem**.
- It is conjectured to be hard.
- If discrete logarithm is easy, then one can solve the Diffie-Hellman problem.
  - Because a and b can then be obtained by Eve.
- But the other direction is still open.

A Parallel History

- Diffie and Hellman's solution to the secret-key agreement problem led to public-key cryptography.
- At around the same time (or earlier) in Britain, the RSA public-key cryptosystem was invented first before the Diffie-Hellman secret-key agreement scheme was.
  – Ellis, Cocks, and Williamson of the Communications Electronics Security Group of the British Government Communications Head Quarters (GCHQ).

Is a forged signature the same sort of thing
as a genuine signature,
or is it a different sort of thing?
— Gilbert Ryle (1900–1976),
The Concept of Mind (1949)

"Katherine, I gave him the code.
He verified the code."
"But did you verify him?"
— *The Numbers Station* (2013)

Digital Signatures[a]

- Alice wants to send Bob a *signed* document x.
- The signature must unmistakably identifies the sender.
- Both Alice and Bob have public and private keys

$$e_{\text{Alice}}, e_{\text{Bob}}, d_{\text{Alice}}, d_{\text{Bob}}.$$

- Every cryptosystem guarantees $D(d, E(e, x)) = x$.
- Assume the cryptosystem also satisfies the commutative property

$$E(e, D(d, x)) = D(d, E(e, x)). \tag{16}$$

  - E.g., the RSA system satisfies it as $(x^d)^e = (x^e)^d$.

[a]Diffie and Hellman (1976).

Digital Signatures Based on Public-Key Systems

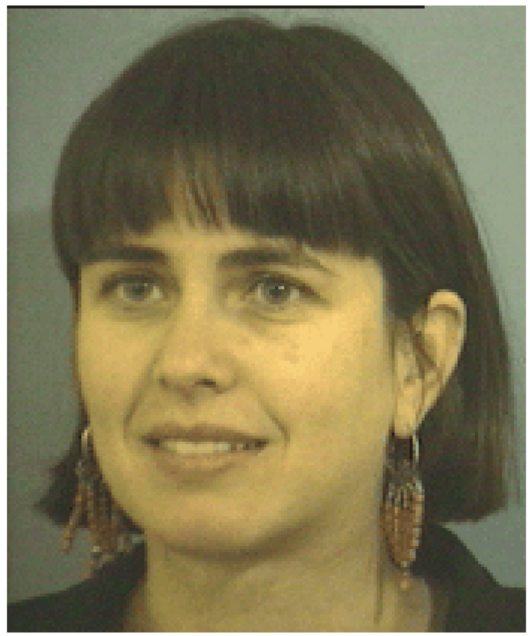
- Alice signs x as
$$(x, D(d_{\text{Alice}}, x)).$$
- Bob receives (x, y) and verifies the signature by checking
$$E(e_{\text{Alice}}, y) = E(e_{\text{Alice}}, D(d_{\text{Alice}}, x)) = x$$
based on Eq. (16).
- The claim of authenticity is founded on the difficulty of inverting E_{Alice} without knowing the key d_{Alice}.

Probabilistic Encryption[a]

- A deterministic cryptosystem can be broken if the plaintext has a distribution that favors the "easy" cases.
- The ability to forge signatures on even a vanishingly small fraction of strings of some length is a security weakness if those strings were the probable ones!
- A scheme may also "leak" *partial* information.
  - Parity of the plaintext, e.g.
- The first solution to the problems of skewed distribution and partial information was based on the QRA.

[a]Goldwasser and Micali (1982). This paper "laid the framework for modern cryptography" (2013).

Shafi Goldwasser[a] (1958–)



[a]Turing Award (2013).

Silvio Micali[a] (1954–)

[a]Turing Award (2013).

Goldwasser and Micali

A Useful Lemma

Lemma 76 *Let $n = pq$ be a product of two distinct primes. Then a number $y \in Z_n^*$ is a quadratic residue modulo n if and only if $(y \,|\, p) = (y \,|\, q) = 1$.*

- The "only if" part:
  - Let x be a solution to $x^2 = y \bmod pq$.
  - Then $x^2 = y \bmod p$ and $x^2 = y \bmod q$ also hold.
  - Hence y is a quadratic modulo p and a quadratic residue modulo q.

The Proof (concluded)

- The "if" part:
  - Let $a_1^2 = y \bmod p$ and $a_2^2 = y \bmod q$.
  - Solve

$$\begin{aligned} x &= a_1 \bmod p, \\ x &= a_2 \bmod q, \end{aligned}$$

    for x with the Chinese remainder theorem.
  - As $x^2 = y \bmod p$, $x^2 = y \bmod q$, and $\gcd(p, q) = 1$, we must have $x^2 = y \bmod pq$.

The Jacobi Symbol and Quadratic Residuacity Test

- The Legendre symbol can be used as a test for quadratic residuacity by Lemma 63 (p. 547).
- Lemma 76 (p. 659) says this is not the case with the Jacobi symbol in general.
- Suppose $n = pq$ is a product of two distinct primes.
- A number $y \in Z_n^*$ with Jacobi symbol $(y \,|\, pq) = 1$ may be a quadratic nonresidue modulo n when

$$(y \,|\, p) = (y \,|\, q) = -1,$$

because $(y \,|\, pq) = (y \,|\, p)(y \,|\, q)$.

The Setup

- Bob publishes $n = pq$, a product of two distinct primes, and a quadratic nonresidue y with Jacobi symbol 1.
- Bob keeps secret the factorization of n.
- Alice wants to send bit string $b_1 b_2 \cdots b_k$ to Bob.
- Alice encrypts the bits by choosing a random quadratic residue modulo n if b_i is 1 and a random quadratic nonresidue (with Jacobi symbol 1) otherwise.
- So a sequence of residues and nonresidues are sent.
- Knowing the factorization of n, Bob can efficiently test quadratic residuacity and thus read the message.

The Protocol for Alice

for $i = 1, 2, \ldots, k$ **do**
  Pick $r \in Z_n^*$ randomly;
  if $b_i = 1$ **then**
    Send $r^2 \bmod n$; {Jacobi symbol is 1.}
  else
    Send $r^2 y \bmod n$; {Jacobi symbol is still 1.}
  end if
end for

The Protocol for Bob

```
for i = 1, 2, ..., k do
    Receive r;
    if (r | p) = 1 and (r | q) = 1 then
        b_i := 1;
    else
        b_i := 0;
    end if
end for
```

Semantic Security

- This encryption scheme is probabilistic.
- There are a large number of different encryptions of a given message.
- One is chosen at random by the sender to represent the message.
- This scheme is both polynomially secure and **semantically secure**.

What Is a Proof?

- A proof convinces a party of a certain claim.
  - "$x^n + y^n \neq z^n$ for all $x, y, z \in \mathbb{Z}^+$ and $n > 2$."
  - "Graph G is Hamiltonian."
  - "$x^p = x \bmod p$ for prime p and $p \not| x$."
- In mathematics, a proof is a fixed sequence of theorems.
  - Think of it as a written examination.
- We will extend a proof to cover a proof *process* by which the validity of the assertion is established.
  - Recall a job interview or an oral examination.

Prover and Verifier

- There are two parties to a proof.
  - The **prover** (**Peggy**).
  - The **verifier** (**Victor**).
- Given an assertion, the prover's goal is to convince the verifier of its validity (**completeness**).
- The verifier's objective is to accept only correct assertions (**soundness**).
- The verifier usually has an easier job than the prover.
- The setup is very much like the Turing test.[a]

---

[a] Turing (1950).

Interactive Proof Systems

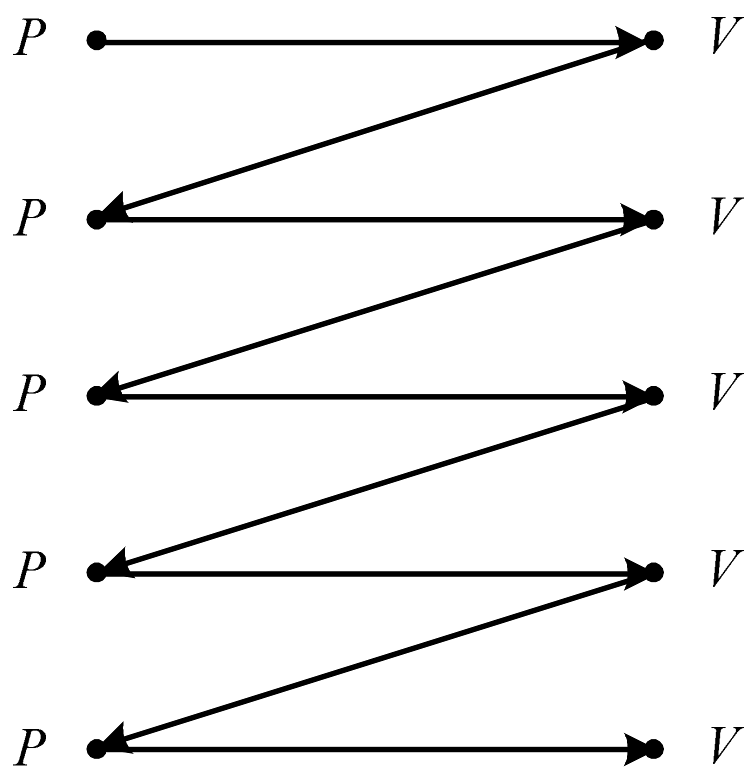
- An **interactive proof** for a language L is a sequence of questions and answers between the two parties.
- At the end of the interaction, the verifier decides whether the claim is true or false.
- The verifier must be a probabilistic polynomial-time algorithm.
- The prover runs an exponential-time algorithm.[a]
  - If the prover is not more powerful than the verifier, no interaction is needed!

[a]See the problem to Note 12.3.7 on p. 296 and Proposition 19.1 on p. 475, both of the textbook, about alternative complexity assumptions without affecting the definition. Contributed by Mr. Young-San Lin (B97902055) and Mr. Chao-Fu Yang (B97902052) on December 18, 2012.

Interactive Proof Systems (concluded)

- The system decides L if the following two conditions hold for any common input x.
  - If $x \in L$, then the probability that x is accepted by the verifier is at least $1 - 2^{-|x|}$.
  - If $x \notin L$, then the probability that x is accepted by the verifier with *any* prover replacing the original prover is at most $2^{-|x|}$.
- Neither the number of rounds nor the lengths of the messages can be more than a polynomial of $|x|$.

An Interactive Proof



IP[a]

- **IP** is the class of all languages decided by an interactive proof system.
- When $x \in L$, the completeness condition can be modified to require that the verifier accepts with certainty without affecting IP.[b]
- Similar things cannot be said of the soundness condition when $x \notin L$.
- Verifier's coin flips can be public.[c]

[a]Goldwasser, Micali, and Rackoff (1985).
[b]Goldreich, Mansour, and Sipser (1987).
[c]Goldwasser and Sipser (1989).

The Relations of IP with Other Classes

- $\mathrm{NP} \subseteq \mathrm{IP}$.
  - IP becomes NP when the verifier is deterministic and there is only one round of interaction.[a]
- $\mathrm{BPP} \subseteq \mathrm{IP}$.
  - IP becomes BPP when the verifier ignores the prover's messages.
- IP actually coincides with PSPACE.[b]

---

[a] Recall Proposition 36 on p. 326.
[b] Shamir (1990).

Graph Isomorphism

- $V_1 = V_2 = \{1, 2, \ldots, n\}$.
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic** if there exists a permutation π on $\{1, 2, \ldots, n\}$ so that $(u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$.
- The task is to answer if $G_1 \cong G_2$.
- No known polynomial-time algorithms.
- The problem is in NP (hence IP).
- It is not likely to be NP-complete.[a]

[a] Schöning (1987).

GRAPH NONISOMORPHISM

- $V_1 = V_2 = \{1, 2, \ldots, n\}$.
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **nonisomorphic** if there exist *no* permutations π on $\{1, 2, \ldots, n\}$ so that $(u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$.
- The task is to answer if $G_1 \not\cong G_2$.
- Again, no known polynomial-time algorithms.
  - It is in coNP, but how about NP or BPP?
  - It is not likely to be coNP-complete.
- Surprisingly, GRAPH NONISOMORPHISM $\in$ IP.[a]

[a] Goldreich, Micali, and Wigderson (1986).

A 2-Round Algorithm

Victor selects a random $i \in \{1, 2\}$;
Victor selects a random permutation π on $\{1, 2, \ldots, n\}$;
Victor applies π on graph G_i to obtain graph H;
Victor sends (G_1, H) to Peggy;
if $G_1 \cong H$ **then**
    Peggy sends $j = 1$ to Victor;
else
    Peggy sends $j = 2$ to Victor;
end if
if $j = i$ **then**
    Victor accepts;
else
    Victor rejects;
end if

Analysis

- Victor runs in probabilistic polynomial time.
- Suppose $G_1 \not\cong G_2$.
  - Peggy is able to tell which G_i is isomorphic to H, so $j = i$.
  - So Victor always accepts.
- Suppose $G_1 \cong G_2$.
  - No matter which i is picked by Victor, Peggy or any prover sees 2 *identical* graphs.
  - Peggy or any prover with exponential power has only probability one half of guessing i correctly.
  - So Victor erroneously accepts with probability $1/2$.
- Repeat the algorithm to obtain the desired probabilities.

Knowledge in Proofs

- Suppose I know a satisfying assignment to a satisfiable boolean expression.
- I can convince Alice of this by giving her the assignment.
- But then I give her more knowledge than is necessary.
  - Alice can claim that she found the assignment!
  - Login authentication faces essentially the same issue.
  - See `www.wired.com/wired/archive/1.05/atm_pr.html` for a famous ATM fraud in the U.S.

Knowledge in Proofs (concluded)

- Suppose I always give Alice random bits.
- Alice extracts no knowledge from me by any measure, but I prove nothing.
- Question 1: Can we design a protocol to convince Alice (the knowledge) of a secret without revealing anything extra?
- Question 2: How to define this idea rigorously?

Zero Knowledge Proofs[a]

An interactive proof protocol (P, V) for language L has the **perfect zero-knowledge** property if:

- For every verifier V', there is an algorithm M with expected polynomial running time.
- M on any input $x \in L$ generates the same probability distribution as the one that can be observed on the communication channel of (P, V') on input x.

[a]Goldwasser, Micali, and Rackoff (1985).

Comments

- Zero knowledge is a property of the prover.
  - It is the robustness of the prover against attempts of the verifier to extract knowledge via interaction.
  - The verifier may deviate arbitrarily (but in polynomial time) from the predetermined program.
  - A verifier cannot use the transcript of the interaction to convince a third-party of the validity of the claim.
  - The proof is hence not transferable.

Comments (continued)

- Whatever a verifier can "learn" from the specified prover P via the communication channel could as well be computed from the verifier alone.
- The verifier does not learn anything except "$x \in L$."
- Zero-knowledge proofs yield no knowledge in the sense that they can be constructed by the verifier who believes the statement, and yet these proofs do convince him.

Comments (continued)

- The "paradox" is resolved by noting that it is *not* the transcript of the conversation that convinces the verifier.
- But the fact that this conversation was held "on line."
- *Computational* zero-knowledge proofs are based on complexity assumptions.
  - M only needs to generate a distribution that is computationally indistinguishable from the verifier's view of the interaction.

Comments (concluded)

- It is known that if one-way functions exist, then zero-knowledge proofs exist for every problem in NP.[a]
- The verifier can be restricted to the honest one (i.e., it follows the protocol).[b]
- The coins can be public.[c]

---

[a] Goldreich, Micali, and Wigderson (1986).
[b] Vadhan (2006).
[c] Vadhan (2006).

Are You Convinced?

- A newspaper commercial for hair-growing products for men.
  - A (for all practical purposes) bald man has a full head of hair after 3 months.
- A TV commercial for weight-loss products.
  - A (by any reasonable measure) overweight woman loses 10 kilograms in 10 weeks.

Quadratic Residuacity

- Let n be a product of two distinct primes.
- Assume extracting the square root of a quadratic residue modulo n is hard without knowing the factors.
- We next present a zero-knowledge proof for the input

$$x \in Z_n^*$$

being a quadratic residue.

Zero-Knowledge Proof of Quadratic Residuacity

1: **for** $m = 1, 2, \ldots, \log_2 n$ **do**
2: Peggy chooses a random $v \in Z_n^*$ and sends $y = v^2 \bmod n$ to Victor;
3: Victor chooses a random bit i and sends it to Peggy;
4: Peggy sends $z = u^i v \bmod n$, where u is a square root of x; $\{u^2 \equiv x \bmod n.\}$
5: Victor checks if $z^2 \equiv x^i y \bmod n$;
6: **end for**
7: Victor accepts x if Line 5 is confirmed every time;

A Useful Corollary of Lemma 76 (p. 659)

Corollary 77 *Let $n = pq$ be a product of two distinct primes. (1) If x and y are both quadratic residues modulo n, then $xy \in Z_n^*$ is a quadratic residue modulo n. (2) If x is a quadratic residue modulo n and y is a quadratic nonresidue modulo n, then $xy \in Z_n^*$ is a quadratic nonresidue modulo n.*

- Suppose x and y are both quadratic residues modulo n.
- Let $x \equiv a^2 \bmod n$ and $y \equiv b^2 \bmod n$.
- Now xy is a quadratic residue as $xy \equiv (ab)^2 \bmod n$.

The Proof (concluded)

- Suppose x is a quadratic residue modulo n and y is a quadratic nonresidue modulo n.
- By Lemma 76 (p. 659), $(x \mid p) = (x \mid q) = 1$ but, say, $(y \mid p) = -1$.
- Now xy is a quadratic nonresidue as $(xy \mid p) = -1$, again by Lemma 76 (p. 659).

Analysis

- Suppose x is a quadratic residue.
  - Then x's square root u can be computed by Peggy.
  - Peggy can answer all challenges.
  - Now,

$$z^2 \equiv \left(u^i\right)^2 v^2 \equiv \left(u^2\right)^i v^2 \equiv x^i y \bmod n.$$

  - So Victor will accept x.

Analysis (continued)

- Suppose x is a quadratic nonresidue.
  - Corollary 77 (p. 687) says if a is a quadratic residue, then xa is a quadratic nonresidue.
  - As y is a quadratic residue, $x^i y$ can be a quadratic residue (see Line 5) only when $i = 0$.
  - Peggy can answer only one of the two possible challenges, when $i = 0$.[a]
  - So Peggy will be caught in any given round with probability one half.

---

[a] Line 5 ($z^2 \equiv x^i y \bmod n$) cannot equate a quadratic residue z^2 with a quadratic nonresidue $x^i y$ when $i = 1$.

Analysis (continued)

- How about the claim of zero knowledge?
- The transcript between Peggy and Victor when x is a quadratic residue can be generated without Peggy!
- Here is how.
- Suppose x is a quadratic residue.[a]
- In each round of interaction with Peggy, the transcript is a triplet (y, i, z).
- We present an efficient Bob that generates (y, i, z) with the same probability *without* accessing Peggy's power.

[a]There is no zero-knowledge requirement when $x \notin L$.

Analysis (concluded)

1: Bob chooses a random $z \in Z_n^*$;

2: Bob chooses a random bit i;

3: Bob calculates $y = z^2 x^{-i} \bmod n$;[a]

4: Bob writes (y, i, z) into the transcript;

---

[a]Recall Line 5 on p. 686: Victor checks if $z^2 \equiv x^i y \bmod n$.

Comments

- Assume x is a quadratic residue.
- For (y, i, z), y is a random quadratic residue, i is a random bit, and z is a random number.
- Bob cheats because (y, i, z) is *not* generated in the same order as in the original transcript.
  - Bob picks Peggy's answer z first.
  - Bob then picks Victor's challenge i.
  - Bob finally patches the transcript.

Comments (concluded)

- So it is not the transcript that convinces Victor, but that *conversation with Peggy is held "on line."*
- The same holds even if the transcript was generated by a cheating Victor's interaction with (honest) Peggy.
- But we skip the details.