Cryptography is ...
♦ Protecting Privacy of Data
♦ Authentication of Identities
♦ Preservation of Integrity

… basically any protocols designed to operate in an environment *absent* of universal trust.
Basic Communication Problem
Eve listening to Alice talking to Bob
Hello

Two-Party Environments
Alice Bob

Remote Coin Flipping
- Alice and Bob decide to make a decision by flipping a coin.
- Alice and Bob are not in the same place.

Ground Rule
Protocol must be asynchronous.
- We cannot assume simultaneous actions.
- Players must take turns.

Is Remote Coin Flipping Possible?
Two-part answer:
- NO – I will sketch a formal proof.
- YES – I will provide an effective protocol.

A Protocol Flow Tree
Completing the Pruning

When the pruning is complete one will end up with either

- a winner before the protocol has begun, or
- a useless infinite game.

Conclusion of Part I

Remote coin flipping is utterly impossible!!!
How to Remotely Flip a Coin

The INTEGERS

0  4  8  12  16 ...
1  5  9 13 17 ...
2  6 10 14 18 ...
3  7 11 15 19 ...

Even

4n + 1:
2  6 10 14 18 ...
3  7 11 15 19 ...

4n - 1:
1  5  9 13 17 ...

Fact 1

Multiplying two (odd) integers of the same type always yields a product of Type +1.

\[(4p+1)(4q+1) = 16pq + 4p + 4q + 1 = 4(4pq + p + q) + 1\]
\[(4p-1)(4q-1) = 16pq - 4p - 4q + 1 = 4(4pq - p - q) + 1\]

Fact 2

There is no known method (other than factoring) to distinguish a product of two “Type +1” integers from a product of two “Type -1” integers.
Fact 3
Factoring large integers is believed to be much harder than multiplying large integers.

How to Remotely Flip a Coin

Alice
- Randomly select a bit $b \in \{\pm 1\}$ and two large integers $P$ and $Q$ – both of type $b$.
- Compute $N = PQ$.
- Send $N$ to Bob.

Bob
- After receiving $N$ from Alice, guess the value of $b$ and send this guess to Alice.

Alice
- Randomly select a bit $b \in \{\pm 1\}$ and two large integers $P$ and $Q$ – both of type $b$.
- Compute $N = PQ$.
- Send $N$ to Bob.
How to Remotely Flip a Coin

Alice
- Randomly select a bit \( b \in \{ \pm 1 \} \) and two large integers \( P \) and \( Q \) – both of type \( b \).
- Compute \( N = PQ \).
- Send \( N \) to Bob.

Bob
- After receiving \( N \) from Alice, guess the value of \( b \) and send this guess to Alice.
- After receiving \( b \) from Bob, reveal \( P \) and \( Q \).

Bob wins if and only if he correctly guesses the value of \( b \).

How to Remotely Flip a Coin

Alice
- Randomly select a bit \( b \in \{ \pm 1 \} \) and two large integers \( P \) and \( Q \) – both of type \( b \).
- Compute \( N = PQ \).
- Send \( N \) to Bob.

Bob
- After receiving \( N \) from Alice, guess the value of \( b \) and send this guess to Alice.

Bob wins if and only if he correctly guesses the value of \( b \).

Does This Work?

There is no known method (other than factoring) to distinguish a “Type +1” product from a “Type –1” product.

\[
(4p+1)(4q+1) = 16pq+4p+4q+1 = 4(4pq+p+q)+1
\]

\[
(4p-1)(4q-1) = 16pq-4p-4q+1 = 4(4pq-p-q)+1
\]

Bob cannot distinguish without factoring.
Can Alice Cheat?
- Randomly pick large integers $p$, $q$, $r$, and $s$.
- Send Bob $N = (4p+1)(4q+1)(4r–1)(4s–1)$.
- If Bob guesses $-1$, send $P = (4p+1)(4q+1)$ and $Q = (4r–1)(4s–1)$.
- If Bob guesses $+1$, send $P = (4p+1)(4r–1)$ and $Q = (4q+1)(4s–1)$.

How to Remotely Flip a Coin

Alice
- Randomly select a bit $b \in \{±1\}$ and two large primes $P$ and $Q$ – both of type $b$.
- Compute $N = PQ$.
- Send $N$ to Bob.

Bob
- After receiving $N$ from Alice, guess the value of $b$ and send this guess to Alice.
- After receiving $b$ from Bob, reveal $P$ and $Q$.

Bob wins if and only if he correctly guesses the value of $b$.

Checking Primality

Basic result from group theory –
If $p$ is a prime, then for integers a such that $0 < a < p$, then $a^{p-1} \mod p = 1$.
This is almost never true when $p$ is composite.

How are the Answers Reconciled?
- The impossibility proof assumed unlimited computational ability.
- The protocol is not 50/50 – Bob has a small advantage.

Applications of Remote Flipping
- Remote Card Playing
- Internet Gambling
- Various “Fair” Agreement Protocols
Bit Commitment
We have implemented remote coin flipping via bit commitment.

Commitment protocols can also be used for:
- Sealed bidding
- Undisclosed contracts
- Authenticated predictions

One-Way Functions
We have implemented bit commitment via one-way functions.

One-way functions can be used for:
- Authentication
- Data integrity
- Strong “randomness”

One-Way Functions
Two basic classes of one-way functions
- Mathematical
  - Multiplication: \( Z = X \cdot Y \)
  - Modular Exponentiation: \( Z = Y^X \mod N \)
- Ugly

The Fundamental Equation
\[ Z = Y^X \mod N \]

When \( Z \) is unknown, it can be efficiently computed.

The Fundamental Equation
When \( X \) is unknown, the problem is known as the discrete logarithm and is generally believed to be hard to solve.
The Fundamental Equation

\[ Z = Y^X \mod N \]

When \( Y \) is unknown, the problem is known as *discrete root finding* and is generally believed to be hard to solve...

...unless the factorization of \( N \) is known.

The problem is not well-studied for the case when \( N \) is unknown.

How to compute \( Y^X \mod N \)

- Compute \( Y^X \) and then reduce \( \mod N \).
  - If \( X, Y, \) and \( N \) each are 1,000-bit integers, \( Y^X \) consists of \( \sim 2^{1010} \) bits.
  - Since there are roughly \( 2^{250} \) particles in the universe, storage is a problem.

- Repeatedly multiplying by \( Y \) (followed each time by a reduction modulo \( N \)) \( X \) times solves the storage problem.
  - However, we would need to perform \( \sim 2^{300} \) 32-bit multiplications per second to complete the computation before the sun burns out.
How to compute $Y^X \mod N$

Multiplication by Repeated Doubling

To compute $X \cdot Y$, compute $Y, 2Y, 4Y, 8Y, 16Y, \ldots$ and sum up those values dictated by the binary representation of $X$.

Example: $26Y = 2Y + 8Y + 16Y$. 
How to compute $Y^X \mod N$

**Exponentiation by Repeated Squaring**

To compute $Y^X$,

- compute $Y, Y^2, Y^4, Y^8, Y^{16}, \ldots$
- and multiply those values dictated by the binary representation of $X$.

**Example:** $Y^{26} = Y^2 \cdot Y^8 \cdot Y^{16}$. 

January 8, 2002

Practical Aspects of Modern Cryptography
How to compute $Y^X \mod N$

We can now perform a 1,000-bit modular exponentiation using ~1,500 1,000-bit modular multiplications.

- 1,000 squarings: $y, y^2, y^4, \ldots, y^{2^{1000}}$
- ~500 “ordinary” multiplications

Large-Integer Operations

- Addition and Subtraction
- Multiplication
- Division and Remainder (Mod N)
- Exponentiation

Large-Integer Addition

```
  123456789
+ 987654321
```

```
  123456789
+ 987654321
  ---------
```

```
  123456789
+ 987654321
  ---------
```

```
  123456789
+ 987654321
  ---------
```
In general, adding two large integers – each consisting of \( n \) small blocks – requires \( O(n) \) small-integer additions.

Large-integer subtraction is similar.
In general, multiplying two large integers – each consisting of $n$ small blocks – requires $O(n^2)$ small-integer multiplications and $O(n)$ large-integer additions.
Careful bookkeeping can save nearly half of the small-integer multiplications (and nearly half of the time).

Recall computing $Y^X \mod N$

- About 2/3 of the multiplications required to compute $Y^X$ are actually squarings.
- Overall, efficient squaring can save about 1/3 of the small multiplications required for modular exponentiation.

Karatsuba Multiplication

$\begin{align*}
(Ax+B)(Cx+D) &= ACx^2 + (AD+BC)x + BD \\
&= 4 \text{ multiplications, 1 addition}
\end{align*}$

Karatsuba Multiplication

$\begin{align*}
(Ax+B)(Cx+D) &= ACx^2 + (AD+BC)x + BD \\
&= 4 \text{ multiplications, 1 addition}
\end{align*}$
Karatsuba Multiplication

\[(Ax+B)(Cx+D) = ACx^2 + (AD+BC)x + BD\]

4 multiplications, 1 addition

\[(A+B)(C+D) = AC + AD + BC + BD\]
Karatsuba Multiplication

\[(Ax+B)(Cx+D) = ACx^2 + (AD+BC)x + BD\]

4 multiplications, 1 addition

\[(A+B)(C+D) = AC + AD + BC + BD\]
\[(A+B)(C+D) - AC - BD = AD + BC\]
3 multiplications, 2 additions, 2 subtractions

January 8, 2002
Practical Aspects of Modern Cryptography
Karatsuba Multiplication

\[(Ax+B)(Cx+D) = ACx^2 + (AD+BC)x + BD\]
4 multiplications, 1 addition

\[(A+B)(C+D) = AC + AD + BC + BD\]
\[(A+B)(C+D) - AC - BD = AD + BC\]
3 multiplications, 2 additions, 2 subtractions

This can be done on integers as well as on polynomials, but it’s not as nice on integers because of carries.

The larger the integers, the larger the benefit.

Modular Reduction

Generally, computing \((A\cdot B) \mod N\) requires much more than twice the time to compute \(A\cdot B\).

Division is slow and cumbersome.
Modular Reduction

Generally, computing \((A \times B) \mod N\) requires much more than twice the time to compute \(A \times B\).

Division is slow and cumbersome.

Division is disgusting.

Division is dreadful.

Division is wretched.
Modular Reduction

Generally, computing \((A \cdot B) \mod N\) requires much more than twice the time to compute \(A \cdot B\).

Division is slow and cumbersome.

The Montgomery Method

The Montgomery Method performs a domain transform to a domain in which the modular reduction operation can be achieved by multiplication and simple truncation. Since a single modular exponentiation requires many modular multiplications and reductions, transforming the arguments is well justified.

Montgomery Multiplication

Let \(A\), \(B\), and \(M\) be \(n\)-block integers represented in base \(x\) with \(0 \leq M < x^n\). Let \(R = x^n\). \(\gcd(R, M) = 1\).

The Montgomery Product of \(A\) and \(B\) modulo \(M\) is the integer \(ABR^{-1} \mod M\).

Let \(M' = -M^{-1} \mod R\) and \(S = ABM' \mod R\).

Fact: \((AB + SM)/R \equiv ABR^{-1} \pmod M\).

Using the Montgomery Product

The Montgomery Product \(ABR^{-1} \mod M\) can be computed in the time required for two ordinary large-integer multiplications.

Montgomery transform: \(A \rightarrow AR \mod M\).

The Montgomery product of \((AR \mod M)\) and \((BR \mod M)\) is \((ABR \mod M)\).

Sliding Window Method

Another way to speed up modular exponentiation is by precomputation of many small products.

For instance, if I have \(y, y^2, y^3, \ldots, y^{15}\) computed in advance, I can multiply by (for example) \(y^{13}\) without having to multiply individually by \(y, y^4, \text{ and } y^8\).

One-Way Functions

\[ Z = Y^X \mod N \]
One-Way Functions

Informally, $F : X \to Y$ is a one-way if

- Given $x, y = F(x)$ is easily computable.
- Given $y$, it is difficult to find any $x$ for which $y = F(x)$.

The family of functions $F_{Y,N}(X) = Y^X \mod N$ is believed to be one-way for most $N$ and $Y$.

No one has ever proven a function to be one-way, and doing so would, at a minimum, yield as a consequence that $P \neq NP$.

When viewed as a two-argument function, the (candidate) one-way function $F_{Y,N}(X) = Y^X \mod N$ also satisfies a useful additional property which has been termed quasi-commutivity: $F(F(Y,X_1),X_2) = F(F(Y,X_2),X_1)$ since $Y^{X_1 X_2} = Y^{X_2 X_1}$.

Diffie-Hellman Key Exchange

Alice
- Randomly select a large integer $a$ and send $A = Y^a \mod N$.

Bob
- Randomly select a large integer $b$ and send $B = Y^b \mod N$. 
**Diffie-Hellman Key Exchange**

**Alice**
- Randomly select a large integer $a$ and send $A = Y^a \mod N$.
- Compute the key $K = B^a \mod N$.

**Bob**
- Randomly select a large integer $b$ and send $B = Y^b \mod N$.
- Compute the key $K = A^b \mod N$.

What does Eve see? $Y, Y^a, Y^b$ … but the exchanged key is $Y^{ab}$.

**Belief:** Given $Y, Y^a, Y^b$, it is difficult to compute $Y^{ab}$.

**Contrast with discrete logarithm assumption:** Given $Y, Y^a$, it is difficult to compute $a$.

---

**Diffie-Hellman Key Exchange**

**Alice**
- Randomly select a large integer $a$ and send $A = Y^a \mod N$.

**Bob**
- Randomly select a large integer $b$ and send $B = Y^b \mod N$.

$B^a = Y^{ba} = Y^{ab} = A^b$

---

**Diffie-Hellman Key Exchange**

**Alice**
- Randomly select a large integer $a$ and send $A = Y^a \mod N$.
- Compute the key $K = B^a \mod N$.

**Bob**
- Randomly select a large integer $b$ and send $B = Y^b \mod N$.
- Compute the key $K = A^b \mod N$.

---

**More on Quasi-Commutivity**

Quasi-commutivity has additional applications.

- decentralized digital signatures
- membership testing
- digital time-stamping
One-Way Trap-Door Functions

\[ Z = Y^X \mod N \]

Recall that this equation is solvable for \( Y \) if the factorization of \( N \) is known, but is believed to be hard otherwise.

RSA Public-Key Cryptosystem

\[ \text{Alice} \]

\[ \text{Anyone} \]

- Select two large random primes \( P \) & \( Q \).
- Publish the product \( N = PQ \).

\[ \text{Alice} \]

\[ \text{Anyone} \]

- Select two large random primes \( P \) & \( Q \).
- Publish the product \( N = PQ \).

To send message \( Y \) to Alice, compute \( Z = Y^X \mod N \).
RSA Public-Key Cryptosystem

Alice

• Select two large random primes P & Q.
• Publish the product N=PQ.

Anyone

• To send message Y to Alice, compute Z=Y^X mod N.
• Send Z and X to Alice.

In practice, the exponent X is almost always fixed to be X = 65537 = 2^{16} + 1.

Some RSA Details

When N=PQ is the product of distinct primes,

\[ Y^X \mod N = Y \]

whenever

\[ X \mod (P-1)(Q-1) = 1 \] and \( 0 \leq Y < N. \)

Alice can easily select integers E and D such that \( E \cdot D \mod (P-1)(Q-1) = 1. \)

Some RSA Details

Encryption: \( E(Y) = Y^E \mod N. \)
Decryption: \( D(Y) = Y^D \mod N. \)

\[
D(E(Y)) \\
= (Y^E \mod N)^D \mod N \\
= Y^{ED} \mod N \\
= Y
\]
RSA Signatures

An additional property

\[ D(E(Y)) = Y^{ED} \mod N = Y \]
\[ E(D(Y)) = Y^{DE} \mod N = Y \]

Only Alice (knowing the factorization of N) knows D. Hence only Alice can compute \( D(Y) = Y^D \mod N \). This \( D(Y) \) serves as Alice’s signature on \( Y \).

Public Key Directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Public Key</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>( N_A )</td>
<td>( E_A(Y) = Y^E \mod N_A )</td>
</tr>
<tr>
<td>Bob</td>
<td>( N_B )</td>
<td>( E_B(Y) = Y^E \mod N_B )</td>
</tr>
<tr>
<td>Carol</td>
<td>( N_C )</td>
<td>( E_C(Y) = Y^E \mod N_C )</td>
</tr>
</tbody>
</table>

(Recall that \( E \) is commonly fixed to be \( E = 65537 \).)

Certificate Authority

“Alice’s public modulus is \( N_A = 331490324840… \)”
-- signed CA.

Trust Chains

Alice certifies Bob’s key.
Bob certifies Carol’s key.
If I trust Alice should I accept Carol’s key?

Authentication

How can I use RSA to authenticate someone’s identity?

If Alice’s public key \( E_A \), just pick a random message \( m \) and send \( E_A(m) \).

If \( m \) comes back, I must be talking to Alice.

Authentication

Should Alice be happy with this method of authentication?

Bob sends Alice the authentication string

\[ y = “I \ owe \ Bob \$1,000,000 \ - \ signed \ Alice.” \]

Alice dutifully authenticates herself by decrypting (putting her signature on) \( y \).
Authentication
What if Alice only returns authentication queries when the decryption has a certain format?

RSA Cautions
Is it reasonable to sign/decrypt something given to you by someone else?
Note that RSA is multiplicative. Can this property be used/abused?

RSA Cautions
\[ D(Y_1) \cdot D(Y_2) = D(Y_1 \cdot Y_2) \]
Thus, if I've decrypted (or signed) \( Y_1 \) and \( Y_2 \), I've also decrypted (or signed) \( Y_1 \cdot Y_2 \).

The Hastad Attack
Given
\[ E_1(x) = x^3 \mod n_1 \]
\[ E_2(x) = x^3 \mod n_2 \]
\[ E_3(x) = x^3 \mod n_3 \]
one can easily compute \( x \).

The Bleichenbacher Attack
PKCS#1 Message Format:
\[
\begin{array}{c}
00 01 XX XX ... XX 00 YY YY ... YY \\
\text{random} \quad \text{message} \\
\text{non-zero} \quad \text{bytes}
\end{array}
\]

“Man-in-the-Middle” Attacks
Alice ↔ Eve ↔ Bob
The Practical Side

- RSA can be used to encrypt any data.
- Public-key (asymmetric) cryptography is very inefficient when compared to traditional private-key (symmetric) cryptography.

For efficiency, one generally uses RSA (or another public-key algorithm) to transmit a private (symmetric) key. The private session key is used to encrypt any subsequent data. Digital signatures are only used to sign a digest of the message.