Secrecy and Trust via Applications of Cryptography

OR

Whom Do You Trust?

Robert Beezer
University of Puget Sound

Introduction

- Cryptography is usually associated with "secret codes" — making them and breaking them. We think of the CIA, KGB, FBI and maybe the NSA.

- However, with the ubiquity of computer networks, cryptography can be just as valuable for allowing individuals to trust people they have never met.

- I hope you will ask, "Just how paranoid is this guy?"

- Should encryption be outlawed?
  Louis Freeh, Former Director of the FBI, thinks so.
  October 2002 Testimony to the US Congress:

  "Robust and commercially available encryption products are proliferating, and no legal means has been provided to law enforcement to deal with this problem, as was recently done by Parliament in the United Kingdom," Freeh said in his testimony. "Terrorists, drug traffickers and criminals have been able to exploit this huge vulnerability in our public safety matrix."

Protocols

A *protocol* is a series of steps designed to allow two (or more) parties to accomplish a task.

An everyday example: placing a custom order from a merchant over the phone.

Protocols we will discuss:

Encryption	Secret Sharing
Authentication	Flipping Coins
“Blind” Notaries	Digital Cash
Time Stamping	

The Players:

Alice	Initiates a protocol
Bob	The other party in the protocol
Eve	An eavesdropper
Mallet	A malicious active attacker
Trent	A trusted authority

Mathematical Functions

Most cryptographic protocols rely on:

- Complicated functions with nice properties
- Basic ideas from probability

There is a lot of good mathematics behind the construction and analysis of these functions.

A good *random number generator* is a must in order to provide the appearance of unpredictability.
Linux starts one up at boot-time.

A *hash function* is a "many-to-one" function that takes a huge set of inputs into a smaller (though possibly still large) set of outputs. Also called a *compression* function.

Functions and protocols use such large numbers that the basics of *probability* make brute-force attacks by Mallet infeasible.

2^{128} operations at the rate of 100 million/second requires 10^{17} years — about 10 million times longer than the age of the universe.

A Hash Function at Work

From John F. Kennedy's Inaugural Address: (2660 bits)

Finally, whether you are citizens of America or citizens of the world, ask of us here the same high standards of strength and sacrifice which we ask of you. With a good conscience our only sure reward, with history the final judge of our deeds; let us go forth to lead the land we love, asking His blessing and His help, but knowing that here on earth God's work must truly be our own.

Hashed: (1032 bits, base 64)

```
cHU1sna+U1y4YHqAcP3Aczg5I5ch2OyB3EaFRsC00Gg
LxfjErgyF+ZOubnPOKVikW2Rv9/9lLWznN3HqGTOXyU
MTOV4/0udE2d/D9d/ACkQI5ZTebqmiH+29ChutOCzoR
meH5Q4Qu5S01CCgewGZfsW/kTtmRDHkS1PQ6youe88
```

Same quote with one bit changed (first F $\rightarrow$ G):

Ginally, whether you are citizens of America or citizens of the world, ask of us here the same high standards of strength and sacrifice which we ask of you. With a good conscience our only sure reward, with history the final judge of our deeds; let us go forth to lead the land we love, asking His blessing and His help, but knowing that here on earth God's work must truly be our own.

Hashed:

```
ZeqZmLXwUXzHAKQwFj5bkluJLiGz9R3ZN6qbzXo3M09
7JoJIBK1D1txkdFeqHn/M2QfXRUA6WP++v+wshajLd3
JFY1VjgGINuZV0PYqStrp6B2xg+Vsr5gRhrRDT8NVVZ
6YFUYzZdlh0JawfjDtKomBu7J+iU8gL+bWAj7GpRrU
```

Encryption Algorithms

- Encryption algorithms disguise a message as much as possible, so that it has the appearance of random noise.
- These *"one-way" functions* require users to provide a *key* as part of the function's input. One-way functions should be difficult to reverse without information about the key(s).

Symmetric algorithms use the same key for both encoding and decoding.

- DES (Data Encryption Standard), the former official US Government standard. Built by IBM with NSA's help, adopted 1976. NSA reduced key size from 112 bits to 56 bits. Rationale behind algorithm never disclosed. Has been successfully attacked (late 90's).
- IDEA (International Data Encryption Algorithm) Built by ETH (Zurich), 1990. Has been subjected to extensive review. No fee for noncommercial use.
- AES (Advanced Encryption Standard), the newly adopted US Government Standard. Based on 4×4 matrices with entries from a finite field of $2^8 = 256$ elements.

Public-key algorithms use a *public key* for encryption and a separate *private key* for decryption. The reversal of the one-way function is difficult even with the public key.

- RSA (Rivest-Shamir-Adelman, 1978) The keys use large prime numbers. The one-way function can be reversed easily if one can find the factorization of a 300 digit number into a product of two 150 digit prime numbers.
- Recently some large numbers ($\approx$ 150 digits) have been factored using numerous workstations and many months or years of time. The current unsolved "RSA Challenge" is to factor a 174 digit integer for a \$10,000 prize. The 309 digit (1024 bit) challenge carries a \$100,000 prize.
- PKP (Public Key Partners) a partnership including Stanford and MIT has held patents on RSA, and several other public-key algorithms, limiting widespread adoption.

Public-Key Cryptography for the Masses

PGP (Pretty Good Privacy) by Phillip Zimmerman is freely available software that uses a mixture of RSA and IDEA. GPG (GnuPrivacyGuard) based on the OpenPGP specification is similar in spirit.

1. Alice randomly generates a “session key.”
2. With the session key Alice encrypts the message using IDEA or some other symmetric algorithm.
3. With Bob’s public key Alice encrypts the session key using RSA.
4. Alice sends the encrypted key and message to Bob.
5. With his private key Bob recovers the session key using RSA.
6. With the session key Bob recovers the message using the symmetric algorithm.

- Zimmerman was investigated by US for exporting encryption software, and accused of patent infringement.
- MIT: `http://web.mit.edu/network/pgp.html`
  International version: `http://www.pgpi.org/`

Encryption at Work

Alice obtains Bob's public key:

```
From   http://buzzard.ups.edu/home.html      (my home page)
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6
mQCNAi5H4bQAAAEEAOzxjp/HCi2jv93rDh/jRiQfrgHHcE4Hj8V3RJnV31dMW8MR
sKPvTQIQHvcIuyqZb2LOd9XBsLRlJEZBY1uwsy4IzbefTc+FgBONczDmNuNo/ADg
ESuGVvkY9jF/dJUlP9Hm4O2olxOR4zBVStHQljLOzhsjxZ5k83iw05YZHXEpAAUR
tCFSb2JlcnQgQS4gQmVlemVyIDxiZWV6ZXJAdXBzLmVkdT4=
=fIch
-----END PGP PUBLIC KEY BLOCK-----
```

Alice's message to Bob (more from JFK):

We observe today not a victory of party but a celebration of freedom, symbolizing an end as well as a beginning, signifying renewal as well as change for I have sworn before you and Almighty God the same solemn oath our forbears prescribed nearly a century and three-quarters ago.

Encrypted with Bob's public key:

```
-----BEGIN PGP MESSAGE-----
Version: 2.6
hIwDeLDTlhkdcSkBBADZi++R3fDQy4sf08O+ODG+rV4QelHECmT8H4s66/5c5p8o
IMdBP8U1o4sX5xtfUUJu/1aQaIHklmvBZtV9ipYREtXIRIm/BtpNgnUxvwhhAKGX
+/XQeacly7pqoS2A8REpSdrafCCIMVLkbh25gPPZ/JZSZH45NuwtYMmPgt3gRaYA
AADnkWr/qr3jNcZyLAPG1Anv+xpOAmXvafo/TLfHquAIyDDfpLJn1LPbHYaOPRyi
xfJPL4sq2xw8U1AyV4fGHFzXvm2ESxFQdnFN1NknwCG+UpvmtxD2zFbBark7VYhb
qKR6+EnF5AGiLxubdiAfSSjhaZyMO6EbmziCacuepgSvHdohEJen8MtCqeDOrGaj
bAXmGiriOlEopbYkql5/dfB8BnpbasFFZ5MtebfGTGDLghe3UGUJXcllfFQWJXg4
meQ28YZn1PIGSVH/gUC6doMXQAUtI+B5CKca425CFh0kIEExiHMEseS9
=NS2B
-----END PGP MESSAGE-----
```

Mallet in the Middle

1. Mallet intercepts Alice's electronic attempts to obtain Bob's public key. He substitutes his own public key.
2. Alice sends Bob a message, unknowingly using *Mallet's* public key.
3. Mallet intercepts the message, and reads it using his private key.
4. Mallet encrypts the message using Bob's public key and sends the message on to Bob.

- Neither Bob nor Alice is aware that the message has been tampered with.
- Use Trent to act as a key server. An example of a key server is `http://www.keyserver.net/en/`. (1658702 keys stored as of March 4, 2003)
- Bob distributes a hash of his key by non-electronic, conventional means (telephone, business card, postings to newsgroups or mailing lists that are archived). PGP calls this hashed key the *fingerprint*.
- A fingerprint appended to a mail message:

```
PGP 2.6 1024-bit public key
Fingerprint: A4 67 92 0B 50 C9 DC E0 7E A9 FE 69 4D 9A 41 7D
```

Authentic Messages?

Do you believe that your electronic mail is **really** originating with the person that is listed in the header?

How can you be sure? How do we identify people we trust?

The unique style of their handwriting?

The sound of their voice?

The look of their face?

The grip of their handshake?

```
X-Sender: president@whitehouse.gov
Date: Fri, 14 Mar 2003 11:20:21 -0800
To: "E. J. Farrell" <farrell@centre.uwi.tt>
From: "George W. Bush" <president@whitehouse.gov>
Subject: I need your help

Ed,

I would like your advice on some foreign policy
questions.  Would you please come to Washington, DC at
your earliest convenience?  I would like to meet with
you face-to-face since I don't trust this secure telephone
line that NSA gave me.  I gotta get PGP real soon.

All the best,
Dubya
```

Authentication with Digital Signatures

1. Alice produces a hash of her message.
2. Alice encrypts her hash with her private key to create a signature, and attaches it to the message.
3. Bob decrypts Alice's signature with her public key to recover the hash.
4. Bob hashes the message to check that it matches the one from the signature.

- Since only Alice can use her private key, and it would be extremely difficult to substitute another message with the same encrypted hash, Bob believes that the message he received did originate with Alice.
- Alice could also have encrypted her signed message (after step 2) using Bob's public key if she wanted to keep her message secret.
- The executable version of PGP is distributed with a signature by Phillip Zimmerman.
- People sign one another's public keys to attest to their validity. There are even "signing parties."

“Blind” Notaries

Sometimes we will want to have a document signed without the signer being able to read the document. This is useful for voting protocols and digital cash. We can think of this as a notarization service being provided by a blind notary.

1. Alice multiplies her document by a number (called the *blinding factor*).
2. Bob signs the unintelligible, blinded document.
3. Alice divides the message by the blinding factor to obtain the original message with Bob’s signature attached and intact.

- This works only if the signature function and multiplication commute with each other.
- The message could say, “Bob owes Alice $1,000,000.”
- Bob is only attesting to the fact that he signed the document (at a certain time).
- This is analogous to a document in an envelope together with a piece of carbon paper.

Timestamping with Trent

Plaintext — Four pages of mathematics written in TeX

```
\documentstyle[12pt]{article}
\begin{document}
\paragraph{Definitions and Notation}  Let $T = (V, B)$ be an
$t-(v, k, \lambda)$ design with $t < k$. That is; $V$ is a
$v$-set, $B$ is a collection of $k$-subsets of $V$ with the
property that every $t$-set from $V$ is contained in exactly

...

isomorphism testing quickly when needed, but I would have
trouble generating an exhaustive list of {\em all} possible
configurations.

Robert Beezer, April 25, 1994
\end{document}
```

The signature:

```
-----BEGIN PGP MESSAGE-----
Version: 2.6

iQCVAgUBLkfukniw05YZHXEpAQGEywQA0saQoebsuvDMFlBFc/c
SRkXh3Lh5UJdkIxlqlht1wP0TzjC3aJCiUmoqE5U2vAoPWL68MJ
ZgHDt7+xbo5Is8HP9Nvv136GtswTq3vsRCvDP8BVvyC4fDPC5a5
liHpeBV//xvcfijEsMPc7Un/6EFPz8WEtQv4iJEi0qWsd4pnmE=
=seB8
-----END PGP MESSAGE-----
```

Trent

- Bellcore, the research arm of the Baby Bells, published the digital signatures of its original works in the unofficial newspaper of record - the *Sunday New York Times*.
- Perhaps professional societies could provide a similar service in order to decide priority disputes.

Sharing a Secret

- Sometimes we might want others to possess a secret of ours, but without them knowing the secret itself.
- Alice has a secret for the success of her family business, and she would like her son, Bob, to have the secret when she dies — but she does not trust him with it while she is alive. She enlists the aid of Trent, the executor of her estate, whom she does not entirely trust either.

1. Alice has a secret S (a string of characters). She generates a random string of the same length, B.
2. Alice does an exclusive-or (XOR, $\oplus$) of B and S to form T, $\quad B \oplus S = T$.
3. Alice gives B to Bob and T to Trent.

- The strings B and T are useless to Bob and Trent individually. T also appears to be random.
- When Alice dies Trent gives Bob T, and Bob exclusive-ors B and T, $\quad B \oplus T = B \oplus B \oplus S = 0 \oplus S = S$.
- This does not stop Bob and Trent from together recovering the secret prior to Alice's death.

Secret Sharing at Work

```
S = 2 cups flour, 1 egg, 1 cup water,
    3 tsp butter, chocolate chips

B = k31zhbkcwp1tix3dfb14corsg1pgfd2s1
    fc,ajgdftbnomifoakqmi  bsy4et

T = lac,szo4h2uift3h2fgagonkqplkhkfj,
    dgwyucil1yt4re4tms34cxi4knhz2
```

- B and T are unintelligible to Bob and Trent, and in fact neither one can tell which has the random string B.

A Matter of Trust

- Alice claims she can predict today the price of a barrel of oil at the end of this month.
- Alice would like to sell her prediction services to Bob.
- Bob would like evidence of Alice's abilities before he pays for this service.
- Alice offers her very accurate "prediction" from last month as evidence.
- Bob is suspicious that Alice has manufactured her prediction after the end of the month.
- Alice refuses to tell Bob this month's prediction, because she does not want Bob to get it for free.
- How can Alice make a prediction that she cannot adjust later, without simultaneously letting Bob having the use of that prediction?
- Trent could be employed to solve this dilemma. But Trent can be a difficult person to find.

Bit Commitment

1. Alice wishes to commit to a single bit, b (or perhaps many bits).
2. Alice creates two lengthy random strings, R_1, R_2.
3. Alice builds the message $R_1 R_2\, b$.
4. Alice hashes this message to form H.
5. Alice sends Bob the hash, H, and R_1. This is the commitment.
6. Later, Alice sends Bob the original message, $R_1 R_2\, b$.

- Bob can check the commitment by seeing that it contains R_1 and that it hashes correctly.
- Bob cannot try test hashs with $b = 0$ and $b = 1$ just after commitment because he does not know R_2.
- Alice is committed to her bit because it would be too hard to build a message $R_1 R_2'\, b'$ with the same hash.

Flipping Coins

Let's flip a coin over the network.

1. Alice chooses a bit at random.
2. Alice sends a message to Bob commiting to this bit.
3. Bob chooses a bit at random and sends it to Alice.
4. Alice sends a message to Bob opening her commitment.
5. If Alice's and Bob's bits match, the coin is heads. If they are different then the coin is tails.

- We can do this repeatedly to generate a random sequence of bits.
- Alice and Bob can build a key in this manner, with the confidence that neither of them is proposing a key with "unusual" properties. If the messages are encrypted, Eve cannot discern the key.
- More involved protocols can be used to deal playing cards over a network.

Anonymous Money Orders

1. Alice prepares 100 money orders, each for \$50.
2. She puts each into an envelope, together with carbon paper.
3. She presents all 100 at the bank. The bank opens 99 of them, checking that they are **really** for \$50 each.
4. Satisfied, the bank signs the last one through the carbon paper and deducts \$50 from Alice's account.
5. Alice takes the money order out of the envelope and to a merchant. The merchant examines the bank's signature.
6. Satisfied, the merchant accepts the money order and then presents the money order at the bank.
7. The bank checks its signature, and credits \$50 to the merchant's account.

Analysis of Anonymous Money Orders

- The odds that Alice can get the bank to sign a \$1,000,000 money order without detection are 1 in 100. There must be a *large* penalty to discourage this. (Jail time?)
- Alice never needs to reveal her identity to the merchant.
- The bank must know Alice when she gets the signed money order, but they cannot trace the spent money order back to her.
- Both Alice and the merchant can copy and spend the money again. The bank will not recognize the multiple uses, nor will they know who it was that recycled the money order.

Digital Cash

1. Alice prepares 100 identical money orders. Each contains:
   (a) The amount — \$50.
   (b) A random uniqueness string, different for each money order.
   (c) 100 copies of information identifying Alice (SSN, Name, Account, etc.) Each of these has been split via the secret sharing protocol in left and right "halves." Alice commits to each of the 200 halves. Note: There are 100 identification pairs on each of the 100 money orders.
2. Alice blinds each of the money orders individually.
3. The bank chooses 99 of Alice's prepared money orders. Alice unblinds each and opens all the commitments. The bank checks the amount, and the uniqueness string, and combines the two halves of each of the identifications.
4. Satisfied, the bank signs the remaining blinded money order and deducts \$50 from Alice's account.
5. Alice unblinds the money order and presents it to a merchant.

6. The merchant verifies the bank's signature, using the bank's public key.
7. Satisfied, the merchant gives Alice a 100 bit selector string. According to this string, Alice opens the commitment on *one* half of each of the 100 identifications.
8. The merchant takes the money order to the bank.
9. The bank checks its signature, and checks to see if the uniqueness string is in its database of spent money orders.
10. Satisfied, the bank credits $50 to the merchant's account and records the uniqueness string and the 100 open halves of the identifications in its database.

Analysis of Digital Cash

- As before, Alice cannot hope to pass a fraudulent money order by the bank.
- Alice cannot tamper with the money order after she leaves the bank, since that would invalidate the bank's signature.
- The money can be spent or deposited twice, but the bank will eventually find out due to their database of spent uniqueness strings.
- Alice cannot spend the money twice, since the two merchants would be unlikely to provide the same selector string (1 in 2^{100}). Where the selectors differ, the bank will eventually get *both* halves of Alices identification. Combining them will provide Alice's identity.
- If the money order is deposited twice by the merchant, the selector strings used will be identical. The merchant cannot fake a second, different selector string since he cannot open Alice's commitments on her identification halves.
- The merchant cannot identify Alice, since he only sees one half of each of her identifications.

- The bank cannot trace Alice's legitimate use of the money order since they cannot see the uniqueness string when Alice gets the money order signed. (It was blinded at the time.) Also, like the merchant, they only see halves of her identifications.
- The money order is independent of physical location. It can be moved through networks at will.
- The bank does not need to be consulted during the transaction (unlike when we use a VISA card today).
- The money orders are not transferable, not divisible.
- Many of the early implementations of digital cash have fallen on hard times (DigiCash, Digital's Milllicent, IBM's MiniPay).

The Perfect Crime

1. Alice kidnaps a baby.
2. Alice prepares 1,000 money orders for $1,000 each.
3. Alice blinds the money orders and sends them to the parents.
4. Alice instructs the parents to have the bank sign all the money orders, and publish the signatures in the newspaper.
5. The parents comply — they pay for and publish the signatures.
6. Alice attaches the signatures to the money orders, unblinds them (in effect making the signatures look different than when they were published) and tries spending 50 of them.
7. If any of the 50 signatures are invalid, Alice keeps the baby and appeals to the parents again, asking for even more money. If all 50 of the signatures work, the baby is released unharmed at a random, quiet location.
8. Alice continues spending more money orders with complete anonymity.

References and WWW Sites

1. Bruce Schneier, *Applied Cryptography*, Wiley, Second Edition, October 1995.

2. Bruce Schneier, *Secrets and Lies*, Wiley, 2000.

3. Wayne Patterson, *Mathematical Cryptology*, Rowan & Littlefield, 1987.

4. James Bamford, *The Puzzle Palace*, Houghton Mifflin, 1982.

5. Simson Garfinkel, *PGP: Pretty Good Privacy* O'Reilly and Associates, January 1995.

6. Pretty Good Privacy.
   `http://web.mit.edu/network/pgp.html`
   `http://www.pgpi.org/`

7. PGP key server at `http://www.keyserver.net/en/`

8. WWW Virtual Library — Cryptography
   `http://world.std.com/~franl/crypto.html`

9. Usenet Newsgroups: `alt.security.pgp`, `talk.politics.crypto`.