Teaching Secure Communication Protocols Using a Game Representation

Dr Leonard G C Hamey
Department of Computing
Macquarie University, NSW, 2109
`len@ics.mq.edu.au`

Abstract

The Security Protocol Game is a highly visual and interactive game for teaching secure data communication protocols. Students use the game to simulate protocols and explore possible attacks against them. The power of the game lies in the representation of secret and public key cryptography. Specifically, the game provides representations for plain text and encrypted messages, message digests, digital signatures and cryptographic keys. Using these representations, students can construct public key certificates and perform multiple encryption, tunnelling and encrypted key transmission. They can simulate a wide range of protocols including authentication, key exchange and blind signature protocols. Application protocols such as Transport Layer Security and Pretty Good Privacy can be simulated in detail. The game clearly reveals the key issues of confidentiality, integrity, authentication and non-repudiation in secure data communications. Used as a small group learning activity, students gain a deep understanding of protocol design and operation issues. The game is suitable for use in tertiary and professional education courses for managers and information technology students at all levels.

Keywords: Computer network, secure communication, cryptography, protocols, digital signature, key exchange, blind signature, man-in-the-middle attack, replay attack, PGP.

1 Introduction

Secure data communication requires two key elements. Cryptographic methods are used to secure the data for transmission and secure communication protocols provide the framework for communication. Data communication can only be secure when adequate cryptographic methods are combined with suitable protocols. Students need to understand both these concepts well in order to properly understand secure data communication. The aim of this paper is to present the Security Protocol Game, a novel learning tool for teaching secure communications protocols.

Students often have difficulty understanding secure communication protocols. Unlike other data communication protocols, secure protocols must be designed with an adversary in mind – an intruder whose intent is to subvert the secure communication and violate the goals of the parties who are communicating. The design of secure communication protocols is largely driven by the need to prevent intrusion. Subtle errors in a protocol may make it vulnerable to attack. Students must understand the possible attacks in order to understand the protocol design.

Students must also be able to identify weaknesses in protocols that provide opportunities for attack. This level of understanding is best provided by giving students the opportunity to interact with a running protocol. We have developed a game that allows students to simulate secure protocols and explore attacks against them. Used in a small group learning situation, the game provides a stimulating environment where students quickly gain a real understanding of secure communication protocols.

The Security Protocol Game provides a simple representation of public key (Diffie and Hellman, 1976) and secret key cryptographic systems and related algorithms. The representation that we will describe below uses coloured envelopes, coloured paper and coloured key tokens to incorporate the key properties of confidentiality, integrity, authentication and non-repudiation into the game. For example, to encrypt a message, a player encloses it in a coloured envelope. This represents the confidentiality provided by encryption. The rules of the game reinforce these properties, preventing a player from accessing a message that has been encrypted unless they hold the appropriate key token.

The idea of using physical representations to explain security protocols is not new. Chaum (1985) uses a representation involving envelopes and rubber stamps to explain blind signature schemes. Bell, Thimbleby, Fellow, Witten and Koblitz (1999) use a representation involving a chain and padlocks to explain Diffie-Hellman key exchange (Diffie and Hellman, 1976) to a non-technical audience. In neither case do the authors attempt to develop a representation that covers the diverse applications of public-key and secret-key cryptographic systems. The Security Protocol Game provides a representation that can be used to study both simple security protocols and real-world secure communication protocols.

2 Games as Learning Tools

Games are a valuable learning tool that has been employed in a variety of ways in computer science education. Reese (2000) uses the challenge of developing programs that play simple games to motivate students to learn interprocess communication mechanisms. Adams

(1998) similarly uses the goal of developing a successful game playing program to motivate students studying object-oriented programming. Ginat (1995) teaches the theoretical concept of loop invariants through simple games such as nim. Students are challenged to develop a winning strategy for the games, and are taught to use loop invariants to both develop and prove their strategies. Ginat argues that the physical objects involved in these simple games help enrich the students' intuition.

Malone (1980) discusses three key characteristics of instructional games. Although his focus is computer games, these same three characteristics are also relevant to instructional games that are not computer based. Malone's first characteristic is a game goal where the attainment of the goal is uncertain. Even in complete information games, such as nim, the outcome may be initially uncertain since the students do not initially realise the complete information nature of the game (Ginat 1995). Malone's second desirable characteristic is fantasy – an environment that is not socially or physically possible. In particular, Malone favours "intrinsic fantasy" in which the player's skills interact with the fantasy environment, as occurs in simulation games. Curiosity is Malone's third desirable characteristic. A game environment that has the appropriate complexity compared to the learner's existing knowledge will stimulate curiosity as the learner seeks to understand the game environment further.

3 Overview of the Game

The Security Protocol Game provides a learning environment for exploring secure communications protocols and the possible attacks against them. Students are divided into groups of 4-6 players, and provided with game kits. Within each group, one student is selected to play each of Alice and Bob, the two communicating parties. Another student is selected to play Gavin, the governing authority and the source of public key certificates. The same student may also take the role of Colin, who represents the copying and computational capabilities of computer systems. The remaining student or students take the role of Trudy the intruder (Tanenbaum 1996).

The students are seated around a table as shown in figure 1. The students select one of the protocols described in the game and a secure data communication scenario to play. In a typical scenario, Alice wishes to purchase computer software from Bob over the Internet using her credit card for payment. The students may choose to simulate the Transport Layer Security protocol (TLS[1]) for this scenario, or other protocols, some of which are vulnerable to various attacks. Messages are passed between Alice, Bob and Gavin via Trudy, allowing her to attempt attacks upon the protocol. The students find this a stimulating group activity as they help each other run the protocol correctly and try to think up ways to subvert it.

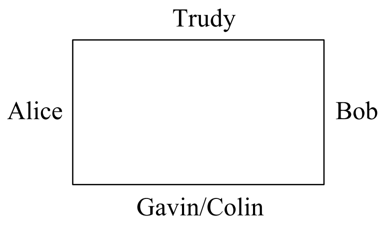


Figure 1: The players seated around a table.

4 Representation in the Game

Each game kit contains a complete set of equipment for simulating protocols that use public key cryptography and/or secret key cryptography. Messages are written on pieces of white paper and passed via Trudy, who may attempt to attack the protocol by creating, modifying or blocking messages.

In order to secure their messages, players may enclose them in an envelope, representing encryption. Only the holder of an appropriate cryptographic key token may open an envelope to access the message inside. Thus, messages can be secured provided that Trudy has not obtained the cryptographic key token. Cryptographic methods may be combined so the players may explore multiple encryption and tunnelling protocols. Key tokens may also be transmitted in messages; the tokens may be secured by enclosing them in envelopes.

The game provides appropriate representations for secret key encryption, public key encryption and private key encryption (figure 2). A message encrypted with a secret key is enclosed in an envelope of the same colour as the secret key token. A player may only open a coloured envelope if they have a key token of the correct colour. This representation captures the privacy provided by the encryption.

A message encrypted with a public key is similarly enclosed in an envelope of the same colour as the public key token. A player must hold the corresponding private key token in order to open the envelope. To help distinguish public-key and secret-key methods, strong colours are used for public key cryptography and pastels are used for secret key cryptography.

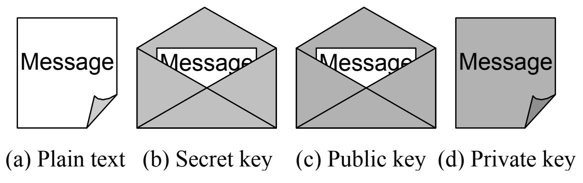


Figure 2: Plain text and encrypted messages. In actual play, the envelopes are closed over the messages.

A message encrypted with a private key is written on a piece of coloured paper matching the colour of the key token. This allows other players to read the message, representing the reasonable assumption that public keys are public information. It also simulates the

[1] TLS is the successor to Secure Socket Layer (SSL), which is commonly used to secure transactions on the world-wide web.

authentication aspect of the encryption – only a holder of the corresponding private key token may write or modify a message on coloured paper. In particular, Gavin is assigned the colour gold, so all players know that messages written on gold paper originate from Gavin. Thus, Gavin can issue public key certificates by writing them on gold paper. Players can also issue their own certificates using their own coloured paper. Figure 3 is an example of a public key certificate issued by Gavin.

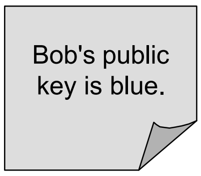


Figure 3: A public key certificate stating that Bob's public key is blue. The certificate is written on gold paper.

The game kits contain addressing labels that are attached to messages (using ordinary paper clips) to indicate their intended routing. These labels simulate the addressing of Internet packets or data segments. Trudy may obey the labels or she may choose to deliver messages incorrectly, possibly substituting the labels. For example, a simple "hello" message from Alice to Bob is shown in figure 4. This message is not secured since it is written on white paper.

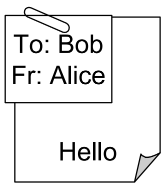


Figure 4: A simple "hello" message from Alice to Bob.

5 Example Scenario

The game contains a variety of scenarios in which secure communications would be useful. In a typical scenario, Alice wishes to purchase computer software over the Internet from Bob, using her credit card for payment. Trudy wishes to subvert the communication for her own benefit or to the detriment of Alice or Bob. In this scenario, Alice and Bob win the game if they are able to securely transmit the credit card number and the software to the other party. Trudy wins if she is able to obtain Alice's credit card number, or obtain a copy of the software without paying for it, or cause Alice to pay double for the software, or cause Alice to receive a corrupted version of the software.

This scenario may be played with a variety of protocols including TLS (Dierks and Allen, 1999). The game kit contains full descriptions of the protocols, including clear instructions on how they are simulated within the game. The students choose a scenario and a protocol to play in each round of the game. In this paper, we will consider a simple protocol that is vulnerable to attack and show how the simulation proceeds. The protocol that we will use is presented in table 1.

In this simulation, we will assume that Alice has a red public key, Bob has a blue public key and Trudy has a green public key. The players do not know with certainty each other's public keys beforehand, so the protocol must provide a method for distributing public key information. Protocol PK2 uses public key certificates to distribute the public keys.

Step	Transmit	Message
1	A→ B	Hello with public key certificate attached
2	B → A	Hello with public key certificate attached
3	A → B	Credit card number encrypted with Bob's public key
4	B → A	Software encrypted with Alice's public key

Table 1: Protocol PK2: A vulnerable protocol for the credit card purchase scenario.

The protocol begins by requiring Alice to initiate contact with Bob. The hello message includes a public key certificate issued by Gavin (written on gold paper) stating the colour of Alice's public key. In this protocol, it is assumed that Alice and Bob have been issued with their own certificates before the scenario commences.

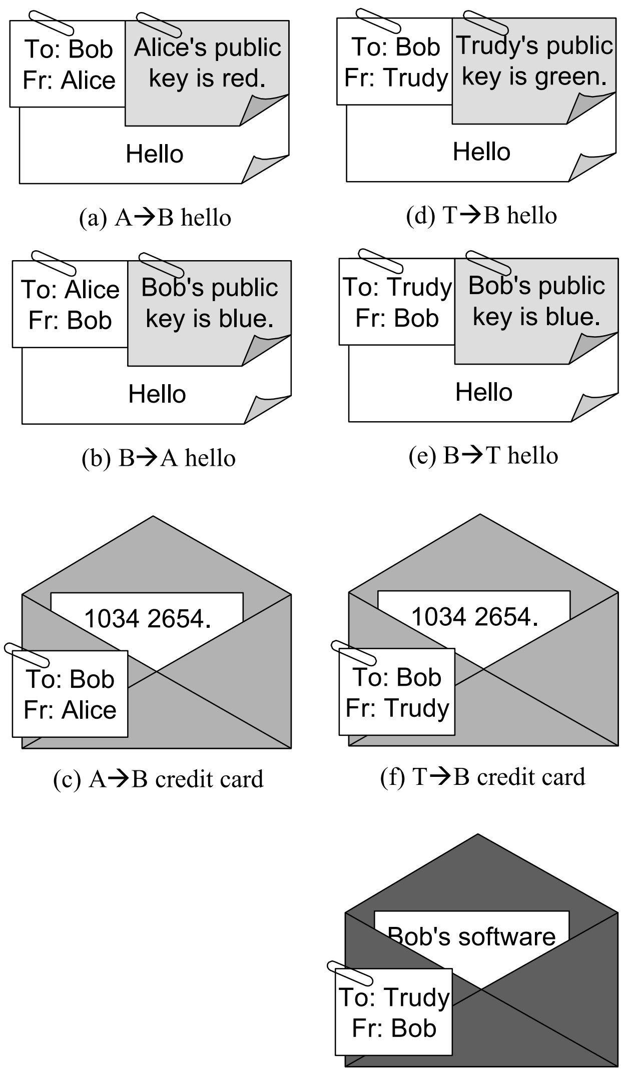
Alice constructs the hello message by writing the message itself on a piece of white paper. Since she has to send a copy of her public key certificate in the message, she asks Colin to make a copy for her. Computer systems can make identical copies of arbitrary data objects – in the game, Colin provides this capability.[2] Colin can copy any game object for any player, including encrypted objects. In this example, Colin copies the certificate onto another piece of gold paper. Alice attaches the copy and an address label to the message and hands it to Trudy. The completed message is shown in figure 5(a).

Trudy now has the opportunity to attempt to attack the protocol. She could modify the hello message itself since it is written on white paper and is not encrypted in any way. She could substitute her own public key certificate for Alice's. However, she cannot modify the certificate, nor can she create a false certificate that states that Alice's public key is green (Trudy's colour). In fact, there is nothing to be gained by modifying this message so Trudy, after considering the possibilities, passes it directly to Bob.

The second step of the protocol requires Bob to respond with a message containing his public key certificate. Since the arriving message satisfies the protocol, Bob

[2] Colin provides a service to the game but is not a participating player in the game. He is like the banker in Monopoly® and is often played by the same person who plays Gavin.

responds accordingly, passing the message to Trudy who again passes it unchanged to Alice. The message is shown in figure 5(b).



(a) A→B hello (d) T→B hello

(b) B→A hello (e) B→T hello

(c) A→B credit card (f) T→B credit card

(g) B→T software

Figure 5: Example scenario – Alice and Bob's messages and Trudy's attack.

Upon receiving Bob's response, Alice proceeds to step 3 of the protocol. She shows Colin the public key certificate and obtains a blue public key token. (Computationally, a public key certificate contains the key itself; a rule of the game allows a player to obtain a key if they hold the corresponding certificate). She writes her credit card number (a number assigned to her as part of setting up the scenario) on a piece of white paper and secures it by enclosing it in a blue envelope – she is entitled to use a blue envelope because she holds a blue public key token. She attaches an address label and hands the message, shown in figure 5(c), to Trudy.

It is at this point that Trudy commences her attack. Several variations are possible on the attack we will describe, and we encourage the reader, if they are not already familiar with the example, to explore the possibilities themselves. We will describe a simple form of attack in which Trudy aborts the connection between Alice and Bob and uses the message she has just received to deceive Bob into sending the software to Trudy.

To develop her attack, Trudy considers the possibility of purchasing the software from Bob using Alice's credit card. In the third step of the protocol, she would need to provide Alice's credit card details to Bob. Trudy observes that the message she has just received is almost exactly the message she would need to send to Bob in the third step of the protocol. Trudy cannot open the message, since it is enclosed in a blue envelope and she does not hold the blue private key token. However, Trudy can modify the addressing information. That is all she needs to do to deceive Bob and obtain a copy of the software for free. Trudy's attack proceeds as follows.

Trudy retains the message she has just received and initiates a new connection with Bob. Since Bob is a vendor of software, he must be willing to sell to any customer, including Trudy, provided that the customer follows the protocol. Trudy therefore constructs a hello message as shown in figure 5(d). This is step 1 of the protocol where Trudy is purchasing the software from Bob.

When Bob receives this message, he checks whether it is valid according to the protocol. The sender's name agrees with the name in the certificate. The certificate is written on gold paper and is valid. Bob therefore responds according to step 2 of the protocol with the message shown in 5(e).

Step 3 of the protocol now requires Trudy to send a message containing a valid credit card number encrypted with Bob's public key. She has the contents of such a message containing Alice's credit card number[3] in the message sent to her by Alice (figure 5(c)). She removes the address label and attaches a new label, creating the message shown in figure 5(f).

When Bob receives the message, he checks the credit card details and finds that they correspond to a valid credit card. Based on this, he encrypts the software with Trudy's public key and sends it to Trudy (figure 5(g)).

At this point, Trudy has successfully subverted the security protocol. She has obtained a free copy of the software and caused Alice to be charged for goods that never arrived. Eventually, Bob and Alice may discover what has happened but Trudy has long since ceased using her network identity "Trudy" and cannot be traced. Thus, protocol PK2 is not secure.

Notice that the security violation we have demonstrated has nothing to do with breaking the cryptographic

[3] Note that the message does not contain Alice's name. In actual use on the Internet, the message would contain the credit card number, card holder name and expiry date, but none of these pieces of information could necessarily be checked against the name found on the public key certificate, since the customer may use a private credit card for a business purchase. If the name on the public key certificate was contained in the message, it would prevent this particular attack.

security of the public-key cryptography system, but rather exploits a weakness in the security protocol. The Security Protocol Game is designed to explore the design and operation of security protocols. It assumes that the cryptographic techniques are secure. Thus, it focuses attention on the strengths and weaknesses of protocols rather than cryptographic systems.

6 Capabilities of the Game

The game can be used effectively to simulate a wide variety of secure communications protocols. The representations of secret key cryptography and public key cryptography reflect capabilities typical of these technologies. In this section we will detail some of the expressive power of the game. Limitations of the game representation will be discussed in a later section.

6.1 Message Digests

A message digest (Needham and Schroeder, 1978; Denning, 1984; Rivest, 1992) is a hash value computed from a message. The digest has two important properties. Firstly, it is not feasible to extract the message from the digest. Secondly, it is not feasible to construct another message that matches the digest. These two properties mean that a message digest can be used to authenticate a message. Message digests are used in digital signature schemes where a digest is encrypted instead of the entire message. A message digest can also be stored with an escrow agent as evidence of the existence of the message at a particular point in time without revealing the contents of the message to the agent. This provides a method of digital time stamping.

In the Security Protocol Game, a message digest is represented as the complete message text written on plain paper with a “no entry” symbol drawn over the top of it. This representation ensures that it is not possible to construct another message that matches the digest. The rules of the game prevent players from modifying message digests, and do not allow the digest to be used in place of the original message even though the digest representation contains the entire message. A message digest, like any message, may be encrypted with a secret key, a public key or a private key. One form of digital signature uses a message digest that has been encrypted with a private key to sign a document (see below).

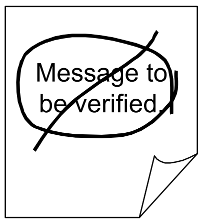


Figure 6: A message digest

6.2 Public Key Certificates

As seen in section 4, the game can simulate public key certificates (Popek and Kline, 1979; Lampson, Abadi, Burrows and Wobber, 1991). Gavin, the certifying authority, issues certificates using his well-known public key (gold) by writing the certificates on gold paper. The players may rely upon these certificates to authenticate public keys to each other. The game can also simulate certificates issued by players other than Gavin, including self-certification where a player issues his own certificates. For example, Bob can issue certificates on blue paper that say “Bob’s public key is blue”.

6.3 Transmitting Encrypted Keys

The Security Protocol Game allows any player to encrypt a key token and transmit it to another party. For example, Alice may generate a session secret key, encrypt it with Bob’s public key and send it to Bob. This technique is used in Pretty Good Privacy (PGP), a protocol for secure email.

In the Security Protocol Game implementation, Alice would obtain a previously unused secret key token from Colin. She would enclose a copy of the token in an envelope the same colour as Bob’s public key, and send it to Bob. Bob can then open the envelope (because he holds the private key) extract the session key and use it for secure secret-key communication with Alice. Figure 7 displays an encrypted session key.

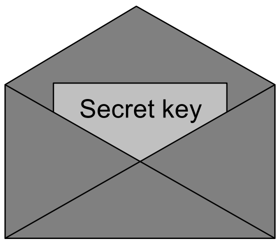


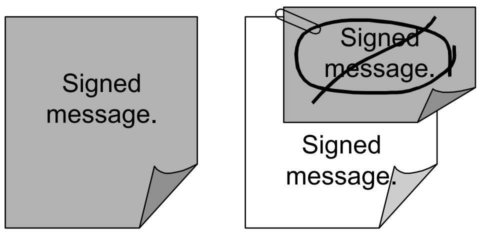
Figure 7: An encrypted session key.

This simple protocol ignores some important issues that must be addressed in practice. Alice must be able to obtain Bob’s public key securely – public key certificates are commonly used to address this issue. Also, the protocol must be secure against replay attacks. RSA key exchange, described in detail below, uses random numbers generated by both Alice and Bob to ensure that the same session key is not reused.

6.4 Digital Signatures

Using public key cryptography, a party may create a digitally signed document by encrypting it with his private key (Diffie and Hellman, 1976). In the Security Protocol Game, a player writes a message on coloured paper to encrypt it with his private key. The coloured paper guarantees the authenticity of the message as long as the private key token remains secure, in just the same way as a digital signature guarantees authenticity as long as the private key remains secure.

Encrypting an entire message with public key methods can be computationally expensive. In an alternative technique for digital signatures, a message digest is computed and encrypted with the private key (Denning, 1984; de Jonge and Chaum, 1987). The digest is attached to the message to digitally sign it. This form of digital signature can also be simulated in the Security Protocol Game. Both representations of digital signatures are shown in figure 8.

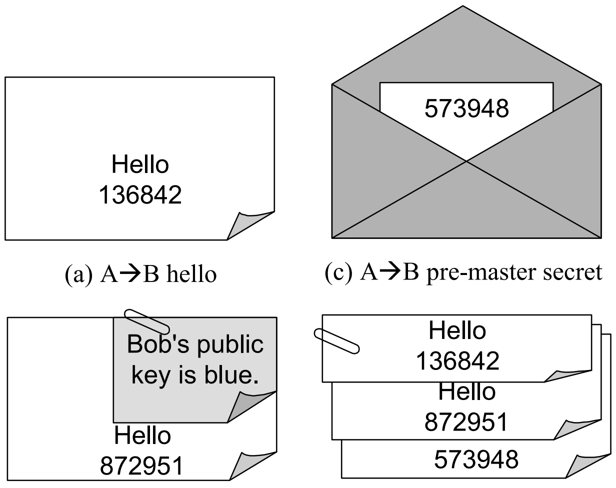


(a) Private-key encrypted message (b) Private-key encrypted digest

Figure 8: Digital signatures

6.5 RSA Key Exchange

One option for establishing a shared secret key in TLS is key exchange using Rivest, Shamir and Adleman's (1978) RSA public-key cryptography. Alice and Bob first exchange session-specific random values, one generated by each party. Bob also sends his public key certificate to Alice. Alice then computes a secret (called the pre-master secret) and transmits it to Bob encrypted with Bob's public key. Bob decrypts the secret using his private key. Both parties then combine the pre-master secret with the session-specific random values. The resulting value is the session key.



(a) A→B hello (c) A→B pre-master secret

(b) B→A hello (d) Final shared information

Figure 9: RSA key exchange.

The simulation of this process in the Security Protocol Game proceeds as follows. The random values are exchanged as plain text messages during the early stages of the protocol, along with Bob's public key certificate. Alice encrypts the pre-master secret with the public key obtained from the certificate and sends it to Bob. Bob extracts the pre-master secret from the message. Both client and server now share the pre-master secret and the two random values. One of the game rules allows a player to show Colin an arbitrary message and receive a secret key corresponding to the message. The first time Colin is shown a particular message, he assigns a new secret key to that message. This represents the computation of a secret key from the message. Thereafter, if other players are able to show Colin the same message, they also obtain the same secret key, representing the same computation. Since both players have the same information, they can both show Colin the same message and obtain the same secret key. Figure 9 shows the three messages exchanged in the protocol and the final shared message that the players use to obtain a shared session key. The addressing labels have been omitted to simplify the figure.

6.6 Pretty Good Privacy

Pretty Good Privacy (PGP) provides security for e-mail. In the PGP protocol (Zimmermann, 1994), Alice digitally signs and encrypts her email before sending it to Bob. The algorithm involves the following steps.

1. Alice computes a digest of the email and encrypts it with her private key. She attaches the encrypted digest to the plain text email to digitally sign the message, similar to figure 8(b).
2. Alice creates a one-time secret key and uses it to encrypt the signed email message.
3. Alice encrypts the one-time key with Bob's public key, which she has previously obtained from his certificate.
4. The encrypted signed email and the encrypted one-time key are sent to Bob. The final message representation in the Security Protocol Game is shown in figure 10.

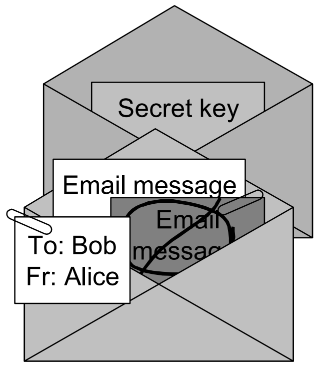


Figure 10: A PGP email message.

6.7 Other Protocols

By now it should be apparent that the Security Protocol Game can simulate a wide variety of protocols. In addition to the examples shown above, the game can be used to simulate TLS (Dierks and Allen, 1999), and protocols based on a Key Distribution Centre including Needham-Schroeder authentication (Needham and Schroeder, 1978), Kerberos tickets (Miller, Neuman, Schiller and Saltzer, 1988), and Otway-Rees authentication (Otway and Rees, 1987). Tanenbaum (1996) describes many of these protocols and possible attacks against them.

6.8 Other Scenarios

In addition to the credit card purchase scenario, the Security Protocol Game can be used to explore other applications of secure data communication including authentication of downloaded software; email security (exemplified by PGP, above), authenticated transactions (such as Alice instructing Bob her stock broker to purchase shares); and virtual private networks.

7 Advanced Concepts

In the preceding sections, we have described security protocols that are based on secret key and public key cryptographic methods, and how these protocols are successfully simulated with the Security Protocol Game. An extension to the representational capabilities of the game provides for simulation of Diffie-Hellman key exchange and blind signatures. These methods rely upon additional properties of the cryptographic algorithms – in particular, commutativity.

Commutativity is a property of algorithms based on exponentiation (such as Rivest, Shamir and Adleman's (1978) RSA algorithm), provided that the same modulus is used for the computations. For example, if Alice defines $E_A(x) = x^a \bmod n$ and Bob defines $E_B(x) = x^b \bmod n$, then $E_A(E_B(m)) = E_B(E_A(m))$ so E_A and E_B are commutative operations. It is clear that the RSA operations E_X (encryption with a public key) and D_X (encryption with the corresponding private key) are commutative for all X; in fact $E_X(D_X(m)) = D_X(E_X(m)) = m$.

In order to simulate commutative operations, we introduce coloured paper clips as a representation of encryption with a public key, and coloured adhesive tags as a representation of digital signing. A plain text message m is folded and sealed with a coloured paper clip to represent encrypting it. A coloured tag is attached to a plain text message to represent applying a digital signature to the message. The same message may be further encrypted with another key by adding another paper clip of a different colour, or an encrypted message may be signed by attaching a coloured tag. The sequence in which the paper clips and tags are added does not affect the appearance of the final message, so this representation accurately captures the commutativity of these operations.

7.1 Diffie-Hellman Key Exchange

In Diffie-Hellman key exchange (Diffie and Hellman, 1976), Alice and Bob initially share a plain text message m and the modulus n. Alice encrypts m with a key that only she knows, obtaining $E_A(m)$ which she transmits to Bob. Bob internally encrypts that message with his own key, obtaining $E_B(E_A(m))$. Bob also encrypts the original message with his key, sending $E_B(m)$ to Alice. Alice then internally encrypts that message obtaining $E_A(E_B(m))$. Because E_A commutes with E_B, Alice and Bob now share a secret that can be used to compute a session secret key.

In the Security Protocol Game, Diffie-Hellman key exchange is simulated using coloured paper clips to represent commutative encryption. Alice and Bob each attach their own coloured paper clip to the plain text message and send it to the other party. When they receive each other's messages, they then attach their own paper clips obtaining identical duplicate messages that can be used to obtain identical secret key tokens from Colin as shown in figure 11.

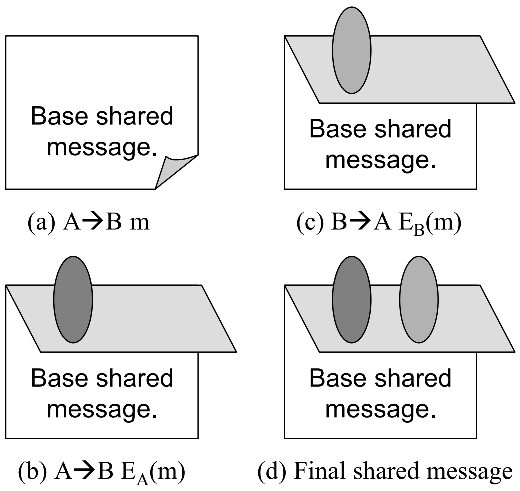


Figure 11: Messages exchanged in Diffie-Hellman key exchange, and the final internally computed shared message.

It is now well known that Diffie-Hellman key exchange is easily broken with a man-in-the-middle attack (Tanenbaum, 1996). That attack can also be readily demonstrated using this representation of the protocol.

7.2 Blind Signatures

In blind signature protocols (Chaum, 1985), Alice sends Bob a message to be signed by Bob. Bob digitally signs the message and returns it. However, Bob is unable to read the message that he is signing because Alice has cryptographically blinded it before sending it to him. Again, commutativity of operations is important, as it must be possible to apply blinding to a plain text message and remove it from a digitally signed message.

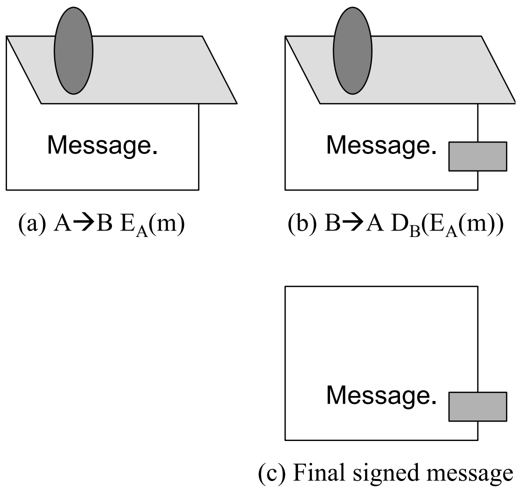


Figure 12: A blind signature scheme.

In the Security Protocol Game representation, Alice encrypts the message by sealing it with a coloured paper clip. Bob applies his signature to the encrypted message by attaching a coloured adhesive tag. When Alice removes the paper clip, she is left with a message with the signature tag still attached. Figure 12 shows this sequence of operations.

The blind signature scheme introduced by Chaum (1985) is more specialised than what we have described above, relying upon the fact that RSA operations are distributive over multiplication. In Chaum's method, Alice generates a random blinding message r and applies Bob's public key E_B to it, obtaining $E_B(r)$. Alice then multiplies this blinding pattern with the message m, sending $E_B(r)m$ to Bob as the message to be signed. Bob applies his private key D_B and returns $D_B(E_B(r)m)$ to Alice. Now $D_B(E_B(r)m)$ is actually $(r^e\ m)^d$ mod n which equals $r^{ed}\ m^d$ mod n, so $D_B(E_B(r)m) = D_B(E_B(r))D_B(m)$; i.e. D_B is distributive over multiplication. Since D_B and E_B are inverses of each other, $D_B(E_B(r)) = r$, so Alice can remove the blinding r from the message she received back, and obtain $D_B(m)$, the signed message. Chaum's technique is stronger in the sense that the signature process remains "blind" whether or not RSA remains secure, since r is random. The representation in figure 12 captures the essential capabilities and benefits of this protocol. It also captures the fact that it is not possible to blind sign more than one message at a time, a property of the blind signature algorithm that is not captured in Chaum's (1985) representation.

8 Limitations of the Game

Within the range of secure communications protocols based upon public and secret key cryptographic techniques, the Security Protocol Game provides a simulation with very few limitations. We have already shown some of the many specific protocols that can be simulated and discussed some of the attacks that can be attempted. The game does, however, have two specific limitations that should be noted. In particular, students specialising in cryptography and data security should be made aware of these limitations when using the game.

The first limitation is that the game does not simulate attacks that rely upon confusion between the tasks of signing and decrypting messages. Two such attacks are possible – misusing a protocol that is designed for digital signing as a decryption engine (Denning, 1984) and misusing a protocol that is designed for authentication to obtain a false digital signature. Both attacks rely upon the fact that a party, such as Bob, uses the same computation D_B to digitally sign a plaintext message m and to decrypt an encrypted message $E_B(m')$. Thus, if an intruder passes $E_B(m')$ to Bob for "signing", they may obtain $D_B(E_B(m'))$ = m' in reply, effectively using Bob as a decryption engine. Alternatively, if the intruder supplies a plain text message m when the protocol requires Bob to decrypt an encrypted message, Bob may reply with $D_B(m)$, effectively signing the message m. Denning (1984) suggests the use of message digests to avoid the first attack. Using separate key pairs for encryption and signing prevents both attacks.

The game does not simulate these protocol misuse attacks. In the game, Bob signs a plain text message by writing it on coloured paper. On the other hand, to decrypt an encrypted message, he removes it from the envelope. The game does not define how Bob would "decrypt" a plain text message or how he would "sign" a message that was already encrypted with his public key. Of course, rules could be added to incorporate these situations into the game. However, we believe that the benefits do not outweigh the added complexity. In essence, the rules that are required to incorporate these attack methods into the game are equivalent to restating the attacks, and we prefer to explain the attacks directly.

A second limitation is that the game does not simulate attacks in which Trudy randomly corrupts encrypted messages. Such attacks may succeed in causing harm to Alice and Bob if the corrupted messages decrypt to well-formed messages at the other end. Because the Security Protocol Game does not allow encrypted messages to be modified by any player not holding an appropriate key, random corruption attacks cannot be simulated. Message Authentication Codes (MACs) or other redundancy techniques are normally employed to prevent these attacks. We may think of the game as simulating protocols in which redundancy is included in all encrypted messages.

9 How to Use the Game

The Security Protocol Game is best used as a small group learning activity. At Macquarie University, we incorporated the game into an undergraduate computer networks course unit. The unit provides an overview of computer networks. It deals with all layers of networking protocols from the physical layer through to application protocols. Special attention is paid to Internet protocols. In this unit, secure data communications occupies only a small number of lecture hours. We have incorporated the Security Protocol Game in two hours of tutorial group activity. Judging by performance on specific questions in examinations, students who play the Security Protocol Game have significantly better understanding of secure communications protocols compared to students in previous unit offerings where the game was not used. A more formal study of the learning benefits of the game is planned in the near future.

The game is very interactive and visual. Students learn how to play it most readily by watching a simple scenario played out. When using the game, we commence the first tutorial hour by demonstrating a simple scenario to the entire class of 20-30 students. One student plays each of Alice, Bob and Trudy while the class tutor runs the game and plays the role of Gavin/Colin, providing instruction and advice to the other players. The game kit includes a simple example protocol that is easily broken and demonstrates the key ideas of the game quickly.

We have found that it is important to introduce the game with this simple example first. The game is moderately complex and students cannot grasp all the representations at one time. The initial example involves only public key cryptography, leaving more advanced concepts such as message digests and key certificates to be explored when the students move on to more sophisticated protocols. This allows the students to rapidly move on to simulating protocols themselves in small groups, while providing additional complexity that unfolds as they move through the game.

The remaining time in the first tutorial hour is spent with the students running the game themselves in groups of four to six students. Each group is supplied with a game kit that included folders for Alice, Bob, Trudy and Gavin/Colin containing all the equipment required by each player. The class tutor provides assistance where needed. The students are encouraged to explore the various protocols that are detailed in the game kit for the credit card purchase scenario. During this time they may encounter key certificates and message digests depending upon their choice of protocols.

In the second tutorial hour, the students have become familiar with the game. At this point, some groups choose to develop their own protocols for other scenarios. For example, one group attempted to develop a protocol for distributing free software over the Internet with authentication. Groups that have not explored the SSL/TLS protocol in the first hour are encouraged to run that protocol through.

We have also used the Security Protocol Game in a postgraduate unit as part of an MBA program. In this unit, the purpose of the game was to give managers a basic understanding of the issues involved in secure data communications. The game was used for one hour with close supervision of the groups. The groups simulated a simple protocol that is easily broken and then chose one other protocol to simulate – in many cases this was the SSL protocol.

In our experience, groups of four to six students can play a typical protocol through in about 20 minutes. A one hour tutorial session is long enough for two to three protocols to be explored in detail. The students find the game very stimulating and discussion among the players ranges from clarification of the rules to high-level strategies for breaking the protocols. It is quite common for Alice or Bob to help Trudy break the protocol.

Due to time limitations in the course units where we have used the game, it has not been possible for students to explore secure data communications as deeply as the game itself would allow. The game includes a variety of scenarios. In one or two hours it is usually only possible for students to explore the credit card purchase scenario in detail. More time would permit the students to explore other applications of secure data communications including virtual private networks, and authenticated distribution of free software (commonly required for web browser plug-ins).

Similarly, students do not have sufficient time to explore the wide variety of protocols that can be simulated by the game. In two hours, the students have time to explore one form of TLS and a number of alternative protocols for the credit card purchase scenario that are vulnerable to attack. More time would allow them to explore the other protocols described above.

We recommend the Security Protocol Game for the following uses:

In an introductory computer networks unit, one to two hours with the security game will allow students to gain an understanding of key concepts such as man-in-the-middle attack, replay attack and the differences between secret key and public key cryptography systems. This level of exposure is also suitable for management students to gain an understanding of the technology issues in secure data communications.

In a unit focussing on security and cryptography, the Security Protocol Game provides a vehicle for explaining the operation of a wide variety of security protocols. Students would benefit from up to five hours using the game in small groups. In addition, the game could be used to help explain key concepts and protocols in lectures.

10 Discussion

The Security Protocol Game is a "complete information" game since, once a protocol is chosen, the information is available to determine whether Trudy can win the game or not. However, the outcome is not obvious to the students since it depends upon whether Trudy is capable of finding a successful attack on the protocol.

In relation to Malone's (1980) three characteristics, the Security Protocol Game has a challenging game goal that engages the students. Especially when the students are playing predefined protocols, the game goal is for Trudy to successfully break the protocol and it is not surprising therefore that even students playing Alice and Bob will offer hints to help Trudy achieve that end. The game also provides an element of intrinsic fantasy. The students are able to explore network interactions that would have financial, social and moral implications in the real world, developing skills that interact with the game environment. The level of curiosity is maintained throughout the game by introducing concepts in stages as the students simulate more sophisticated protocols.

11 Conclusion

The Security Protocol Game is a powerful, highly visual and interactive tool for teaching students the principles and operation of secure data communication protocols. It is ideal for small group learning environments where students explore protocols, attempting to break them by means of well-known attacks. The game uses simple rules and techniques to simulate both public key and secret key cryptography systems. Using these, players can simulate digital signatures, public key certificates, transmission of encrypted keys and digital envelopes, key exchange protocols, key distribution centre protocols and blind signing. Students can attempt a variety of attacks including man-in-the-middle, replay, and denial of service. The game can be used to simulate specific protocols such as TLS and PGP. The game clearly reveals the issues of confidentiality, integrity, authentication and non-repudiation in secure communication protocols.

The game is suitable for use in tertiary and professional education courses for managers and information technology students at all levels. For those students specialising in cryptography and data security, the game provides a clear representation for understanding many important protocols. A complete set of rules and instructions for the Security Protocol Game is available from the author.

12 References

ADAMS, J.C. (1998): Chance-It: An Object-Oriented Capstone Project for CS-1. *ACM SIGCSE Bulletin*, **30**(1): 10-14.

BELL, T., THIMBLEBY, H., FELLOWS, M., WITTEN, I., and KOBLITZ, N. (1999): Explaining cryptographic systems to the general public. In *First IFIP World Conference on Information Security Education (WISE), Proceedings IFIP TC11 WG11.8 Conference.* 221-233. YNGSTGRÖM, L. and FISCHER-HÜBNER, S. (eds), Stockholm University/Royal Institute of Technology, Sweden.

CHAUM, D. (1985): Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, **28**(10): 1030-1044.

DE JONGE, W. and CHAUM, D. (1987): Some variations on RSA signatures and their security. In *Advances in Cryptology – CRYPTO '86 Proceedings*, 49-59. EDLYZKO, A.M. (ed), New York: Springer Verlag.

DENNING, D.E. (1984) Digital signatures with RSA and other public-key cryptosystems. *Communications of the ACM*, **27**(4): 388-392.

DIFFIE, W. and HELLMAN, M.E. (1976): New directions in cryptography. *IEEE Transactions on Information Theory*, **22**(6): 644-654.

DIERKS, T. and ALLEN, C. (1999): *RFC 2246: The TLS Protocol Version 1.0*. Internet Engineering Task Force.

GINAT, D. (1995): Loop Invariants and Mathematical Games. *ACM SIGCSE Bulletin*, **27**(1): 263-267.

LAMPSON, B., ABADI, N., BURROWS, M. and WOBBER, E. (1991): Authentication in distributed systems: theory and practice. *ACM Transactions on Computer Systems*, **10**(4): 265-310.

MALONE, T. (1980): What Makes Things Fun to Learn? Heuristics for Designing Instructional Computer Games. In *Proceedings of the 3rd ACM SIGSMALL Symposium*, 162-169. Palo Alto, CA, USA.

MILLER, S., NEUMAN, B., SCHILLER, J. and SALTZER, J. (1988): Kerberos Authentication and Authorization System. *Section E.2.1 Project Athena Technical Plan*, MIT Athena, Camridge MA.

NEEDHAM, R.M. and SCHROEDER, M.D. (1978): Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, **21**(12): 993-999.

OTWAY, D. and REES, O. (1987): Efficient and timely mutual authentication. *ACM SIGOPS Operating Systems Review*, **21**(1): 8-10.

POPEK, G.J. and KLINE, S.K. (1979): Encryption and secure computer networks. *ACM Computing Surveys*, **11**(4): 331-356.

REESE, D.S. (2000): Using Multiplayer Games to Teach Interprocess Communication Mechanisms. *ACM SIGCSE Bulletin*, **32**(4): 45-47.

RIVEST, R.L., SHAMIR, A. and ADLEMAN, L. (1978): A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, **21**(2): 120-126.

RIVEST, R.L. (1992): *RFC 1321: The MD5 message-digest algorithm*. Internet Engineering Task Force.

SCHNEIER, B. (1995): *E-Mail Security: How to Keep Your Electronic Messages Private*. New York, John Wiley & Sons.

TANENBAUM, A.S. (1996): *Computer Networks* (3rd edition), London, Prentice-Hall.

ZIMMERMANN, P. (1994): *PGP User's Guide*, in Schneier (1995).

13 Acknowledgements

Thanks to Prof. Vijay Varadharajan for suggesting the use of paper clips to simulate Diffie-Hellman key exchange, and for encouraging us to write this paper.