Uncoercible Communication and Deniable Encryption, or How to Lie With Impunity

Matthew Kerner

3/4/2006

1 Introduction

Use of some cryptographic protocols implies a commitment to the data operated on by those protocols. For example, a digital signature may ensure the property of nonrepudiation: the signer would be unable to claim that the signed document is a forgery. Similarly, encrypting data can form a kind of commitment to the data: the decryption process reveals the committed plaintext.

In some scenarios, commitment to a plaintext may be undesirable. For example, in an election setting, we would like to prevent voters from being able to generate proof of their votes, ensuring that they cannot be coerced into voting a certain way, or rewarded for doing so. A secret agent may wish to be able to communicate with her sponsoring organization freely even if a coercive adversary is applying pressure to have her send a particular message. A corporate employee may wish to act as a whistle-blower without fear of management reprisal for having disclosed sensitive information.

A variety of techniques have been developed in recent years to provide various cryptographic services, including privacy, without implying a commitment to the plaintext, even under coercive circumstances. We examine a variety of techniques in this vein.

But first, we should establish some informal definitions.

Coercion occurs when one party forces another, through physical threats, monetary or other rewards, legal action, or other pressure to follow certain behavior. Coercion can occur before, during or after a transaction of interest. Multiple parties involved in a transaction may be coerced, by the same or different adversaries, and each may be asked to follow different behavior.

Uncoercible protocols specify a faking algorithm which the coerced party may follow under coercion, which makes it impossible for her to prove to the adversary that she has followed the coercive instructions, or equivalently, to provide the coercer with a forged receipt indicating compliance which is indistinguishable from an actual non-forged receipt.

Since uncoercible protocols make it impossible to prove something to the adversary, the coerced party has several options.

- Convince the adversary that nothing within the protocol framework can be proven, and so discourage the adversary from coercion in the first place.
- Lie to the adversary by presenting a false result which is indistinguishable from the desired truthful result to the adversary.
- Tell the truth to the adversary, without being able to prove that the tendered result is actually true.

In each of these cases, since the protocol used in communication is uncoercible, the adversary gains no certain advantage from coercion, and may as well not perform it.

Uncoercible protocols require the properties above to be achievable when the coercer has access to the subject before and after the transaction. Deniable protocols require the same only when the coercer has access to the subject afterwards.

2 Techniques

2.1 Uncoercible Communication

[BT94] examines a one-way communication mechanism for use in a hostile environment which ensures that a sender coerced before and/or after the transaction can provide a coercer – who is assumed to have access to the transmitted ciphertext – a plausible receipt for the communication which indicates that the forged plaintext of the sender's choice was the actual plaintext in the communication. Furthermore, the legitimate recipient is guaranteed to be able to detect if a message was sent under coercion with a configurable level of confidence. The motivation for this model, at least in part, was to develop a cryptography-based election system that would be coercion-free.

The model for coercion assumes that the coercer knows the communication algorithm used by both the sender and receiver, has access to the sender before and after the transmission of the message in question, and has access to the communication channel used to send the ciphertext to the receiver. Coercion can occur before the fact (e.g. "You'd better send this message or else!") or after the fact (e.g. "You better have sent this message!" or "You better tell me what you sent!"). However, the model assumes that the coercer doesn't have access to the sender during the act of sending (or such a coercer could manually verify the message that is being sent as it is typed and so detect the sender's attempt to avoid following the coercer's instructions). An active adversary may specify the plaintext that the sender must send. A passive adversary may simply want to obtain the plaintext that the sender chose to send. Either adversary may also request a receipt that can be used with the observed ciphertext to validate that the sender's assertion of the plaintext is plausible.

The model also assumes that the sender and receiver each have shared access to a stream of random private bits. This could be implemented as a private channel from receiver to sender, or more practically with synchronized, identical physical devices which generate a cryptographically random stream of bits (e.g. SecurID).

The basic algorithm presented in [BT94] is as follows. The sender takes message M and encrypts it with a one-time-pad K drawn from the shared sender-receiver private bitstream. K must be long enough to accomodate the length of message M (say L bits), and to also include N additional bits, where N is a predefined security parameter. The ciphertext C consists of the bits of M XORed with the first L bits of K, and with the remaining N bits of K appended, unchanged, to the XORed bits.

When the receiver gets C, she checks the last N bits of the message, which should be identical to the last N bits of K. If this message tail matches, then the message is assumed to be uncoerced, and M can be extracted by XORing the first L bits of C with the first L bits of K.

In the scenario where the coercer provides the sender the message that must be sent, [BT94] models the coercer's instructions as a function $f(\cdot)$ which accepts K as input and outputs a set of constraints on the bits to be sent over the wire (note that the coercer may provide a well-defined function that accepts bits as input and outputs an answer in bits, or a loosely-defined function such as a set of verbal instructions to send a message conveying a certain idea). If f allows the sender to supply any of the last N bits of the ciphertext, she picks the bits to not match the last N bits of the true K (to indicate the coercion to the receiver). If not, she simply sends the message specified by f. In this case, to mislead the recipient into believing that the message was not coerced, the coercer would have to guess the actual N-bit tail of K, which she can do with probability 2^{-N}. Thus, for sufficiently large N, the receiver has high confidence in detecting a coerced message.

In all coercion scenarios, the sender can provide the coercer with a receipt containing some K' which can be used to decrypt C, and which will yield the message of the sender's choice. K' can easily be computed by the sender by XORing the first L bits of C with the message of the sender's choice, and taking the last N bits of C. Since the coercer doesn't have access to the true K, the coercer has no way to distinguish any fake K' from the actual key, and the best the coercer can do is hope that the receipt provided by the sender is legitimate: the sender can lie with impunity.

[BT94] also defines a serial protocol, similar to the one mentioned above, but which allows the recipient to transmit the next bit of K to the sender each time the previous bit of the ciphertext has been sent from the sender to the recipient. This scheme provides some additional desirable properties, such as a way to communicate some uncoerced information in an otherwise coerced message, but the conditions required by this model (a one-way, private channel from receiver to sender) are unlikely to exist in practice.

The model in the serial protocol still assumes that the coercer has no access to the sender at the time of sending. However, suppose that f was designed to record its inputs and outputs, and that these outputs are recorded in such a way that it is difficult for the sender to understand or alter those outputs. The coercer may demand to see the private output of f after the fact, and use it to examine the sender's behavior: if the sender tried passing f different values for a given bit from K (to see whether the coercer is allowing the sender to control this output bit or not based on changing the key bit), and it appeared that the sender had tried to use the faking protocol, then the coercer has some reason to believe that the sender is lying. This type of f diverges from the model slightly, by giving the coercer read-only access after the fact to part of the sender's activity at sending time, but it is still a concern. This type of f would effectively force the protocol to the model where all bits are fixed by the coercer, removing the ability for the sender to leverage the serial protocol to her advantage.

Note that the uncoercible communication model is based on the assumption that it is impossible for the adversary to determine the actual K, and on the adversary's inability to determine whether a given K could possibly be generated by the source used by the sender and receiver to generate it. In practice, these requirements become requirements for the device supplying the shared private bits used by the sender and receiver for the one-time pad.

2.2 Deniable Encryption

[CDNO97] was developed in response to government efforts to systematically gain the ability to access plaintext in encryption schemes ostensibly for law-enforcement purposes. We diverge briefly from the technique described in [CDNO97] to examine this background. The US & British governments proposed various schemes in the late 90s for key escrow for use in all broadly-used encryption cryptosystems. Through a combination of legal mandates and market incentives, these governments encouraged corporations who develop and deploy cryptosystems to provide means for law enforcement to surreptitiously and swiftly gain access to encryption private keys used by anyone named in a wiretap warrant, similar to the manner in which law enforcement can tap phone lines without notifying the tapped party. The cryptographic community responded in various ways (see [AAB$^+$98]), including the development of a variety of deniable encryption schemes, which would allow users of the protocols to plausibly lie about the plaintext if forced to give up their key in court.

[CDNO97] assumes no coercion before the transaction, but allows the sender to plausibly claim a forged plaintext after the fact. It uses a mechanism called a trapdoor permutation to build a system of public-key deniable encryption. The sender chooses an element either randomly or pseudo-randomly, and only the receiver, who knows the trapdoor can feasibly distinguish between the two. The example trapdoor permutations are complex, involving sets of computationally indistinguishable elements, called translucent sets, and predicates to operate on the sets to determine whether an example is a member of a set using some secret information.

In the basic public-key scheme, a separate encryption is used to transmit each bit. To transmit a 1, a random element (represented by a bit-stream) is chosen from the translucent set (the mechanism for constructing an element of the set being the public key). To transmit a 0, a pseudo-random element is chosen instead. The recipient has a secret that allows her to test set membership efficiently, which allows her to decrypt. If challenged by an adversary, the receiver can claim that a 1 was really a 0, but not vice versa. It is possible to build up a more complex scheme that allows the recipient to lie in either direction by sending a set of elements for each bit, and determining the bit based on the parity of the number of random elements in the set, each of which can be lied about.

The authors also also propose a shared-key scheme which is similar to the mechanism in [BT94]. They also suggest a mechanism where multiple messages are encrypted with multiple keys, concatenated, and the sender and receiver have prearranged the choice of key to use to decrypt the real message. Under coercion, they could claim that a different key was used, corresponding to an innocuous plaintext.

2.3 Chaffing and Winnowing

[Riv] proposes a communication model that avoids traditional encryption methods, and uses only authentication techniques in order to achieve message privacy. Unlike normal methods that achieve message privacy, this model was designed to specifically avoid cryptographic primitives associated with government key escrow efforts, so that messages transmitted with it would be immune from key recovery. In fact, the key required for a keyed HMAC function is typically a user's signing or authentication key, which is not used for encryption, and is extremely sensitive for nonrepudiation purposes. As a result, key recovery on this key would be unacceptably risky for anyone to implement, and would only be required to satisfy law enforcement requirements - almost never to satisfy consumer requirements for key escrow.

In [Riv], a message M is decomposed into individual bits. Each bit is packaged into a packet containing a sequence number, the bit itself, and a keyed HMAC value of the first two elements with a private symmetric key. The sender transmits the tuple of

$$(\mathrm{SeqNo}, b, \mathrm{HMAC}(K, \mathrm{SeqNo}, b))$$

(for each bit b with key K). The receiver authenticates the packed using the keyed HMAC before accepting the message. If the HMAC doesn't match, the receiver simply throws the message away as it must have been spoofed. Each packet is sent in the clear, and the only service provided by the protocol is authentication: packets sent by a bogus source who doesn't possess K can be identified and discarded.

Now, the protocol is enhanced to add message privacy, termed "chaffing": for each bit b, two packets are generated and sent with the same sequence number: one as described above, and another that contains $\bar{b}$ and a randomly generated value instead of the legitimate HMAC. The receiver will ignore the packets containing the incorrect hash values ("winnowing"), and will thus be able to reconstruct M. An adversary listening to the conversation will be unable to distinguish the packets containing the correct HMAC values from the incorrect HMAC values without the key: the communication channel now ensures privacy.

The cost of this scheme is in the overhead required to send a message: each bit of the message requires two packets to be generated, sent and examined at the receiver side. In order to reduce the overhead requirements, [Riv] proposes a variation on the protocol involving an "All-or-nothing Transformation" or package transform, which is a keyless operation that operates on the full message M to generate another message M', any part of which conveys no information about M on its own (i.e. *all* of M' is required to learn *anything* about M). Then M' is broken into some reasonable number of blocks, which are each transmitted according to the protocol mentioned above, but replacing bits b with blocks B, and adding only a few chaff packets overall rather than one chaff packet for each legitimate packet. Now the adversary must guess the exact subset of valid packets to get M because of the package transform. This requires $2^N - 1$ tries in the worst case for N packets, which is prohibitively expensive for a reasonably large N. [BB00] proposes some enhancements to the scheme which work around some weaknesses of the package transform by combining the AONT algorithm with some per-bit chaffing/winnowing as described above.

The bitwise version of chaffing and winnowing can easily be extended to provide a form of deniable encryption. The sender now multiplexes p different bitstreams, each with its own unique key K_i to the recipient (i.e. there are $2p$ packets sent for each sequence number - half legitimate for each of the p K_is, and half fake). The recipient assembles the single bitstream that matches the required key, and ignores the rest. However, if a coercer demands a secret key from the sender to "open" the private message stream, the sender can choose which of the p K_is to reveal, allowing the sender to choose between p different plaintexts. Thus the sender could plausibly deny having sent a particular message, either by claiming that the other MAC values were random trash values, or by claiming that the other keys were the fake keys.

2.4 Compulsion Resistant Anonymous Communication

[DC] proposes a network of peer-to-peer nodes which implement an anonymous communication system that allows users to plausibly deny being the originator of a message. This system uses a series of mixes to implement source-based anonymous routing. Only the sender knows who the recipient is: each mix along the way only knows the next hop, and the recipient must use an opaque reply block to send a response back to the anonymous sender, again through a sequence of mixes with only local knowledge.

For a message M, the sender wraps the message in a series of symmetric session key encryptions. For example, to send a message to receiver E, A might send

$$(B, \{C, \{D, \{E\}_{K_D}\}_{K_C}\}_{K_B}, \{\{\{M\}_{K_E}\}_{K_D}\}_{K_C}\}_{K_B})$$

The first element of the tuple is the address of the next recipient. The second is the route determined by the sender, with wrapped encryptions keyed for the sequence of nodes on the route. The third parameter is the message, which is similarly wrapped. Each recipient feeds the message to an internal mix, which batches several received messages, decrypts the latter two elements, sorts the messages to avoid ordering attacks, and sends the messages on to the next recipient. M contains a reply block, which can be used by the recipient to respond to the anonymous sender. The reply block is very similar to the second element of the tuple above, but it is ordered to originate at the recipient and end at the sender, and it need not follow the same route as the initial message. In order to avoid detection by a passive global adversary (one who can observe all network traffic), the message in the reply is recursively encrypted by each mix with before it reaches the destination, where it is decrypted by the original source.

There are some other aspects of this design which are important: packets are padded to obfuscate the length of the route, reply packets appear identical to original message packets, and packets which have been tampered with will be discarded in such a way that an attacker cannot distinguish between a packet discarded due to tampering, a packet discarded because it was a decoy, or a packet discarded because it reached its final destination.

In [DC], an adversary who is able to coerce every node in the route for a particular packet is able to determine the packet's sender or recipient. In this model of coercion, the adversary might force the nodes to relinquish their session keys so that the coercer can find the route. A node might have already deleted an old session key, but if the coercer is fast and forceful enough, she might be able to retrieve the session key for an existing session. There is no reason why the encryption scheme used to protect the plaintext couldn't be an of the uncoercible/deniable category.

The authors move on to show how the protocol can be strengthened to make coercion more difficult, and to provide plausible deniability to the sender. The first modification is to include some multicast steps when messages leave the mixes. This would force an adversary to coerce every recipient of the multicast, and their subsequent next hop nodes, and so on. While this modification doesn't ultimately prevent coercion, it does increase the cost of coercion, and it also forces the coercer to become well known across the network (presumably other parties who fear coercion could decide to lay low when a coercer is active).

The next modification is to design a “compulsion trap”, which is a loop in the route which delivers a message to the message sender twice. A coercer would therefore have to visit the message sender twice: after the first visit, the message sender could destroy all of the state associated with the message so that the coercer couldn't gain anything useful the second time she visits.

Finally, a sender can structure a reply message so that it appears to be routed through several more nodes after arriving at the sender. If all nodes followed this protocol, the message could have plausibly been destined for any node on the route, and a coercer would not be able to determine which node was the true recipient. In order to avoid statistical attacks over multiple messages and multiple rounds of recursion, nodes can choose to include a set of "friends" on each route, along with some other randomly sampled nodes, so that it appears equally likely that any of the "friends" was the other endpoint over time.

2.5 Steganography

Steganography is the field of information hiding. For example, [AP98] mentions a scheme in which a sender and receiver encode a message in the low-order bits of the pixels in an image. Many steganographic systems rely on security by obscurity to ensure message privacy: if an attacker knows the steganographic mechanism used in a particular case, she can extract the plaintext with trivial effort. Other steganographic systems use a key to encode the message. In this case, it would be difficult for the attacker to extract the message even knowing the mechanism. For example, one could agree on a private stream cipher key, and use the stream cipher output to determine the set of pixels in an image whose low-order bits will convey a message.

There are three general approaches to using steganography to achieve plausible deniability, in order of security (note that all three could simultaneously be used):

- Assuming the adversary doesn't know the steganographic method, send a message embedded in a plausible cover text, and if coerced, claim that the cover text was the actual message.
- Use multiple steganographic techniques to encode various messages within a single cover text, and if coerced, reveal some or all of the different techniques, each of whose plaintexts are equally plausible.
- Use a single keyed steganographic technique with multiple keys and plaintexts simultaneously in a single stego-object, and if coerced, reveal some or all of the keys, each of whose plaintexts are equally plausible.

Ideally, steganography would successfully hide the use of encryption in the first place, averting coercion that way.

[CDa] describes a steganographic method which is meant primarily to hide the fact that encryption is being used for a particular message. However, it also uses a variation on the second and third approaches to provide plausible deniability: by encoding multiple ciphertexts within a set of purportedly random bits, it allows a user to reveal some or all of those ciphertexts, each of which is equally plausible.

3 Practical Considerations & Discussion

The common theme across all of these techniques is that the algorithms used provide some degree of freedom to the sender and receiver, usually in a secret key or local randomness, which allows the sender to present multiple plausible explanations for the ciphertext. In order to be able to exploit such degrees of freedom, there are several practical considerations that come into play. First, it is critical to use one's technique of choice consistently, even for innocuous communication. If a coercer has reason to believe that a committing model of communication is accessible to the sender of a message, she will insist that that system be used. If however, an uncoercible or deniable system is the only system that is ever used, then the coercer has no choice but to agree to the use of such a system, or the recipient will become suspicious when the regular routine is broken.

Second, most of the techniques described above do not provide anonymity. The coercer is assumed to know that the sender and receiver are communicating. These techniques only provide assurances regarding the content of the messages. If one wishes to mask the fact that communication is happening, one must use a technique that provides traffic security.

Third, if coercion occurs, and both the sender and receiver are compromised, they must keep their stories straight, or the coercer will know that at least one of them is lying. In practice this is a difficult goal to achieve, and it is therefore critical for the sender and recipient to work out their course of action beforehand.

The distinction between coercion before the fact and after the fact is critical. In the former case, the sender cannot claim to forget the information that the coercer has instructed her to remember in advance: that in itself could be established up front as grounds for a reprisal by the coercer. Instead, the sender must be unable to prove anything to the coercer, and may therefore put forth any claim about the action taken with equal plausibility. Stripping everything else away, this is the scenario that must be supported by an incoercible communications system.

Coercion after the fact is a different matter entirely. One always has the option of simply deleting the information desired by the coercer each time it is generated. The coercer can't punish someone for deleting information if that is the sender's standard practice. One need not even delete the information: one may just claim to have deleted or forgotten it, and there's nothing that the adversary can do as long as it can't be proven that the information exists and is accessible to the coerced party. To a purist, the practice of deleting information is the simplest choice that a user can make if she fears coercion after the fact.

However, [CDNO97] proposes several reasons why it might be better, in practical situations, to avoid deleting the information, and to still have plausible deniability: one might have a legal or policy-based obligation to retain certain information, one might want to retain the information because it is so valuable that its value outstrips the risks associated with coercion, or one might be afraid that claiming to have forgotten or deleted the information would make a coercer angry or more suspicious. In the end, only the message sender can decide whether these considerations merit not simply deleting the information: it may be that deletion is not the best option in some scenarios. The fact remains that for most of us, coercion of the type envisioned by these techniques would be so unpleasant that we would likely spill the beans without a second thought. Thus the advice given in [CDb] is good: "But before you lie... Decide: How far would you go?”

References

[AAB$^+$98] Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, and Bruce Schneier. The risks of key recovery, key escrow & trusted third party encryption: A report by an ad hoc group of cryptographers and computer scientists. *Digital Issues*, 3, 1998.

[AP98] Ross J. Anderson and Fabien A. P. Petitcolas. On the limits of steganography. *IEEE Journal of Selected Areas in Communications*, 16(4):474–481, May 1998. Special issue on copyright & privacy protection.

[BB00] Mihir Bellare and Alexandra Boldyreva. The security of chaffing and winnowing. *Lecture Notes in Computer Science*, 1976:517–??, 2000.

[BT94] J. Benaloh and D. Tuinstra. Uncoercible communication, 1994.

[CDa] Mark Chapman and George Davida. Plausible deniability using automated linguistic stegonagraphy.

[CDb] R. Clayton and G. Danezis. Chaffinch: Plausible deniability for the masses. Computer Laboratory, University of Cambridge.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. *Lecture Notes in Computer Science*, 1294:90–104, 1997.

[CG96] Ran Canetti and Rosario Gennaro. Incoercible multiparty computation (extended abstract). In *IEEE Symposium on Foundations of Computer Science*, pages 504–513, 1996.

[DC] George Danezis and Jolyon Clulow. Compulsion resistant anonymous communications.

[PAK99] Fabien A. P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Information hiding — A survey. *Proceedings of the IEEE*, 87(7):1062–1078, 1999.

[Riv] Ronald L. Rivest. Chaffing and winnowing: Confidentiality without encryption.