Protocols for data hiding in pseudo-random state

Scott Craver, Enping Li, and Jun Yu
Binghamton University, Binghamton NY, USA

ABSTRACT

An emerging form of steganographic communication uses ciphertext to replace the output of a random or strong pseudo-random number generator. PRNG-driven media, for example computer animated backdrops in video-conferencing channels, can then be used as a covert channel, if the PRNG bits that generated a piece of content can be estimated by the recipient.

However, all bits sent over such a channel must be computationally indistinguishable from i.i.d. coin flips. Ciphertext messages and even key exchange datagrams are easily shaped to match this distribution; however, when placing these messages into a continous stream of PRNG bits, the sender is unable to provide synchronization markers, metadata, or error correction to ensure the message's location and proper decoding.

In this paper we explore methods for message transmission and steganographic key exchange in such a "coin flip" channel. We establish that key exchange is generally not possible in this channel if an adversary possesses even a modest noise budget. If the warden is not vigilant in adding noise, however, communication is very simple.

1. INTRODUCTION

Steganography has often focused on embedding messages within an existing cover, which is considered innocuous and is thus allowed for transmission in a censored environent. The primary challenge of conventional steganography is altering a natural cover in a manner that can encode a long message, but without leaving any subtle evidence of tampering that can be used to distinguish covers from stego-objects. A secondary challenge, that of robustness to a warden's noise budget, only exacerbates this problem by requiring a stronger transmission or a smaller payload.

A different approach to covert communication is to seek out innocuous forms of communication that are easy to alter without changing their statistical character. One example of this is concealment in pseudo-random state: if a form of innocent communication possesses a component that is generated in party from random or pseudo-random data—key nonces, activity in online games, computer animations—there is the potential to replace that random data with ciphertext. Indeed, the first published "subliminal channel" entailed embedding of bits in protocol elements, specifically in the random selection of one of four plaintexts in the Rabin cryptosystem.[1] More recent efforts have embedded messages in poker games[2] and Go,[3] guiding the random choice of players to transmit data. However, this is only undetectable if the player was normally random in behavior. Our effort focuses on the random generators guiding computer-generated graphics, which can be used in videoconferencing applications.

In these cases, the cover data is not imperceptibly amended; rather it is partially *generated* from ciphertext. This is ideal if innocent content is normally generated from random data indistinguishable from ciphertext; this completely circumvents the problem of indetectably tampering with natural data, because the generated content is simply natural content with a different RNG or PRNG source. Essentially, Alice and Bob are transmitting random strings that are "wrapped" in cover data by a generation algorithm, and unwrapped by estimating the PRNG bits from the cover data.

This creates a curious type of channel in which Alice and Bob can transmit with impunity, but all of their transmitted bits must be indistinguishable from the output of a random number generator. A warden, Wendy, has access to the transmitted bits, including the ability to occasionally inject an error; and and she will catch Alice and Bob if any statistical test can distinguish the allegedly random bits from truly random bits. This creates uniqe challenges for communication, because this restriction disallows naive error correction, and impedes key exchange.

This research was supported by AFOSR grant FA9550-07-1-0044.

1.1 PRNG-state channels in videoconferencing

The data hiding application that motivates this study exploits the leakage of PRNG bits in novelty backdrops in Apple iChat, a popular instant messaging program with video chat functionality.[4] iChat allows users to superimpose themselves over images, video clips or computer animations; computer animations are often driven by pseudo-random bits, and this state can often be estimated from the rendered animation by a recipient. If PRNG state is replaced with ciphertext, a backdrop animation can be used as a covert channel.

Figure 1 shows an example backdrop animation. We replaced its pseudo-random number generator with a patch that fetches PRNG bits from an external server. This server is then configured to provide ciphertext instead of PRNG bits, effectively leaking a message through the backdrop. Figure 2 shows an animation in use in an iChat session.

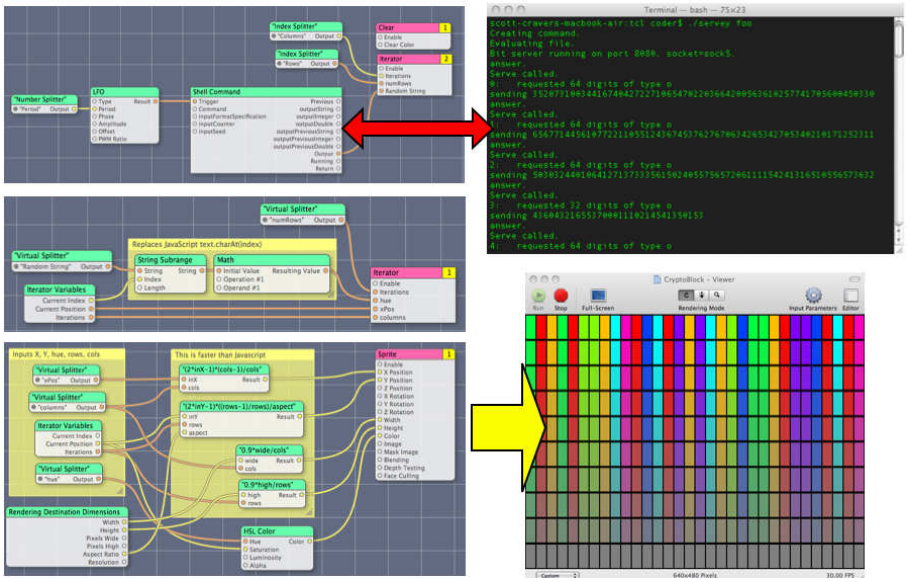

Figure 1. A contaminated animated backdrop, that leaks ciphertext from a server as pseudo-random state governing column hues

Figure 2. An iChat session with a contaminated backdrop animation. Ciphertext (or PRNG state) is leaked through position of raindrops.

There are two reasons to employ such a channel. First, it can be highly robust to an active warden, since bits are encoded as perceptually significant content. Second, this channel is difficult to steganalyze, because the embedding replaces a PRNG state rather than tampering with natural data. If the original PRNG is cryptographically strong, and the ciphertext is properly encrypted, Wendy cannot distinguish between cover and stego animations. We can achieve bitrates on the order of hundreds of bits per second, allowing key exchange in a reasonable amount of time.

In this paper we explore methods for transmission, error correction, and key exchange under this type of channel: a transmission of alterable random bits on the order of hundreds of bits per second, robust but not completely ruling out the injection of an occasional bit error by Wendy. This paper is organized as follows: in section 2 we outline the theoretical model for this type of channel; in section 3 we outline basic primitives for transmission with a secret key; in section 4 we show theoretical limits on unkeyed transmission, for example key exchange, in the presence of an active warden; and in section 5 we provide a key exchange protocol.

2. THE COIN FLIP CHANNEL

The *coin flip channel* is a theoretical model for steganographic communication in supraliminal or mimic-function channels. The concept of a "mimic function" was introduced by Wayner in ,[5] and achieves steganographic communication by automatic generation of covertext, rather than amendment of existing covertext. A channel is thus represented by a generator that translates a bit string to an allowable cover, and a hash that converts a cover to a bit string.

So-called supraliminal channels are outlined in,[6] and also transmit bits by the automatic generation of covers. What distinguishes a "supraliminal" channel from an ordinary mimic function is that the transmitted bits must be robust to alteration by an active warden. For example, a very short message may be encoded as the subject matter of an image, which remains unaltered despite the addition of a great deal of noise.

The chief engineering problem of these channels is developing an automatic generation function whose output resembles natural content. This can be achieved by restricting ourselves to a specific class of natural communication whose content is partially artificial in the first place; one such example is explained below.

We can also achieve supraliminal communication via an ordinary keyed steganographic embedding algorithm, and a perceptual hash, using the algorithm shown in.[4] Here, we choose a pair of covers $\{A, B\}$, and steganographically embed a message in A using the hash of A as a key. The cover A is sent first, so that a warden does not possess the embedding key until after it is received. This technique rests on the assumption that an ideal robust embedding method exists, as well as a robust perceptual hash algorithm.

In any case, we consider a channel in which random strings are converted to and from plausible covers for transmission, so that covert communication is no longer a process of embedding but a process of shaping disallowed messages as allowed messages. Assuming the chief engineering problem is solved (and in specific cases, we argue that it is,) the next problem is a restriction on the channel's use: the transmitted bits must have no observable meaning, or in fact any observable statistical deviation from the "channel noise" that results from feeding truly natural covers into the decoding algorithm. This is the basis for the coin-flip channel model: Alice and Bob are now allowed to send arbitrary strings, but they can only be random.

2.1 The channel model

Assuming that algorithms exist to convincingly disguise *i.i.d.* bits as covers, this generation can be abstracted from the channel model, leaving us with the transmission of the random inputs and outputs. In the coin flip channel, Alice and Bob are emitters of iid uniform bits: each broadcasts a continual sequence $\{A_i\}$ and $\{B_i\}$ of coin flips. They may alter any of their own bits prior to broadcast, but the bits must remain computationally indistinguishable from a sequence of fair coin flips. Wendy the warden monitors all parties' bit streams for any evidence of channel content.

If Alice and Bob are to use these broadcast channels for communication, they must immerse messages that resemble channel noise. Any data with identifiable structure may not be used; ciphertext and appropriately random or pseudo-random strings are allowed, if one can show that they are computationally indistinguishable from channel noise.

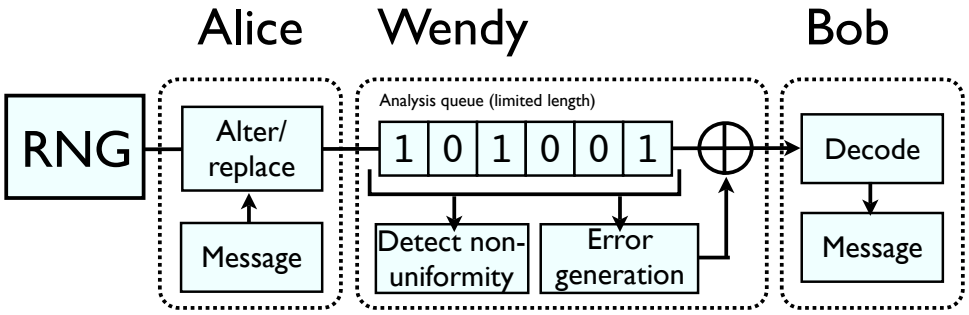

Figure 3. A diagram of the coin flip channel.

Wendy is an ϵ-active warden, able to damage a fraction ϵ of coin flip data. In our channel, ϵ is small, because the data hiding takes place in a *supraliminal* channel, a component of the cover data with semantic meaning that the warden can rarely alter. If Wendy is unable to determine if a message is present, she will still focus her noise budget in places where she suspects messages may be present.

For our purposes, we assume that bits are transmitted in small blocks of m bits each time step. In the steganographic framework we use as a backdrop for this problem, a fixed number of bits are sent per frame in a videoconferencing session, commonly $m < 100$. We also assume that each participant saves the full transmission history of the session; in our case the transmission is on the order of hundreds of bits per second, whose storage does not induce a large strain on either client.

While Wendy can inject errors with impunity up to her noise budget, she is usually restricted by a time limitation: she can delay a small queue of bits from Alice to Bob when deciding where and how to add errors. This allows a form of robustness to error, since Alice can embed data that requires information from a future transmission to read.

3. TRANSMISSION WITH A COIN-FLIP CHANNEL

Given the channel above, Alice and Bob must determine a means to perform key exchange and then communicate securely, all under the restriction that all transmission be indistinguishable from *i.i.d.* uniform random bits. We will show that this is fairly simple once Alice and Bob exchange a key, but key exchange is hard.

The requirement of random appearance requires us to rethink basic aspects of how we communicate. For example, we cannot fit a message with error correction in a straightforward way: the Warden will simply observe that the bits form valid codewords, and are clearly some form of content. Likewise, Alice doesn't have an obvious means to alert Bob to the presence of a message on the channel: wherever she places her message in the data stream, it cannot be marked with any observable delimiter.

It may seem that key exchange is easy over such a channel because key exchange datagrams are often random to begin with. Indeed, we can easily employ a key exchange whose messages are genuinely *i.i.d.* using method described in.[4] However, key datagrams are also fragile to even a single bit error.

3.1 Error correction and synchronization

Since the channel is a continual data stream, Alice needs a means to immerse her message on the channel, so that Bob can locate it. Again, no overt markers or error correction are allowed. Assuming that Alice and Bob agree on a means to exchange datagrams that resemble i.i.d.uniform bits, Wendy can use her noise budget to target bits that she suspects may be message blocks or metadata blocks. Metadata is a valuable target, because it is needed to locate and properly decode embedded messages. We highlight several specific cases:

- *Wendy may target potential error correction.* Given that a specific type of error correction is in use, Wendy will place errors in the stream to maximize the decoding error per block. This is possible if she knows the coding method and how blocks are framed in the data stream.

- *Wendy may target potential synchronization markers.* If Alice and Bob use markers to delimit datagrams, Wendy can target suspected markers to prevent datagram delivery.
- *Wendy may target cryptographic primitives known to be fragile.* If Alice and Bob must exhange public keys, for example, Wendy will try to corrupt at least one bit of each key to render them useless.
- *If suspicious that a protocol may be underway, Wendy may target specific steps.* For example, if Wendy suspects that Alice has transmitted a challenge, she may focus on Bob to destroy a suspected response.

To prevent these problems, datagrams sent over the channel must be robust to targeted error, but also Wendy must be limited in her ability to guess the position of structural elements of the transmission. Furthermore, any key exchange protocol should be free of obvious information flows, e.g.challenge-response protocols, that would allow an attacker to target the location of likely public-key primitives.

3.2 Exploiting the time domain for robustness

To circumvent an active warden we apply a general approach we call *post block, ergo propter block*. In this approach, datagrams are encrypted, marked up, and fit with error correction using methods that are kept secret until after transmission. Once an annotated datagram is immersed in a channel and transmission is confirmed, parameters needed to identify and decode the data are then made available in a short beacon message. The beacon and encoding methods must be plausibly deniable as random data.

The main goal of post block ergo propter block is reducing Wendy's ability to attack potential metadata. Metadata is well-hidden, until a single block is given that points into it. If Wendy guesses the location of this block and damages it, another block can be sent without divulging evidence of communication.

If Alice and Bob share a secret key in advance, Alice can send a message using an ex-post-facto beacon. Before transmission, Alice and Bob both compute n beacon values by encrypting the messages $B_k =$`BEACON::A:B:k` for $1 \leq k \leq n$. Bob compares each block B of bits against the set of beacon strings, and declares a block a beacon if: $|B \oplus B_k| < h$, for a fixed Hamming distance h; and if no block after B_k has been previously seen in the stream.

When Alice wants to send a message, she first immmerses an encrypted datagram into the channel, encrypted using the shared key in CTR mode with block B_k as an initialization vector. Once the transmission is past, Alice immerses the beacon block $B_k|\texttt{Encrypt}_{loc,len}$, containing the beacon marker, followed by a pointer to the message and its length, encrypted in CTR mode using $B_k - 1$ as the initial counter.

For error correction, Alice and Bob can employ any coding method they wish, prior to encryption in CTR mode. CTR mode uses a block cipher as a stream cipher, so a single-bit error in the ciphertext induces a single-bit error in the underlying plaintext.[7] Any structural elements of the coding are concealed by the overlaid encryption. Alice can also provide error correction information in the beacon message, after observing which bits have survived transmission.

Thus, with a secret key, Alice and Bob can employ error correction and delimit messages on the channel using beacons that have significance only to them. Wendy can derail transmission by attacking the beacon or subsequent metadata, but she does not know where this metadata is placed. Unfortunately, this all requires a key exchange, and none of these techniques for coding or delimiting can be used in a key-exchange scenario.

4. LIMITS OF PUBLIC TRANSMISSION

Suppose that Alice and Bob do not initially share a key, and must exchange one using public-key methods. This means that they must transmit key datagrams in the clear, readable by Wendy, but disguised as channel noise. Alice and Bob have the advantage that with the right algorithm, public keys can be uniformly generated; however, they face the disadvantage that public key datagrams are fragile, and the lack of any structure or meaning to the data prevents a damaged datagram from being repaired.

The problem, then, is to represent a random key exchange datagram as a bit string so that it resembles *i.i.d.* bits, but is immune to an ϵ-active warden. There are several possible degrees of resemblance:

- A strongly secure transmission is genuinely uniform: every possible n-bit string sent by Alice has probability 2^{-n}.
- A weakly, or computationally secure transmission cannot be distinguished from a uniform n-bit string. If Alice sends Bob ciphertext properly encrypted with a strong cipher and secret key, the message is non-uniform but weakly secure.

In this section we show that strongly secure transmission is impossible under an active warden—that is, if a warden can inject even a small number of bit errors, all secure coding algorithms of reasonable bitrate are fragile.

4.1 Transmission under an active warden with full knowledge

Because there is no secret key guiding the transmission, we may assume that the warden has full knowledge of the coding method used. Thus Wendy can seek out the most fragile bits when choosing how to spend a noise budget. Whatever algorithm is used, we can regard the set of n-bit strings that decode to the same message as a codeword cell, and thus the algorithm as an (n, m) code—except that, instead of sending a codeword, Alice sends a string chosen uniformly from the codeword's cell. The question is, what (n, m) code corrects at least $n\epsilon$ errors on average?

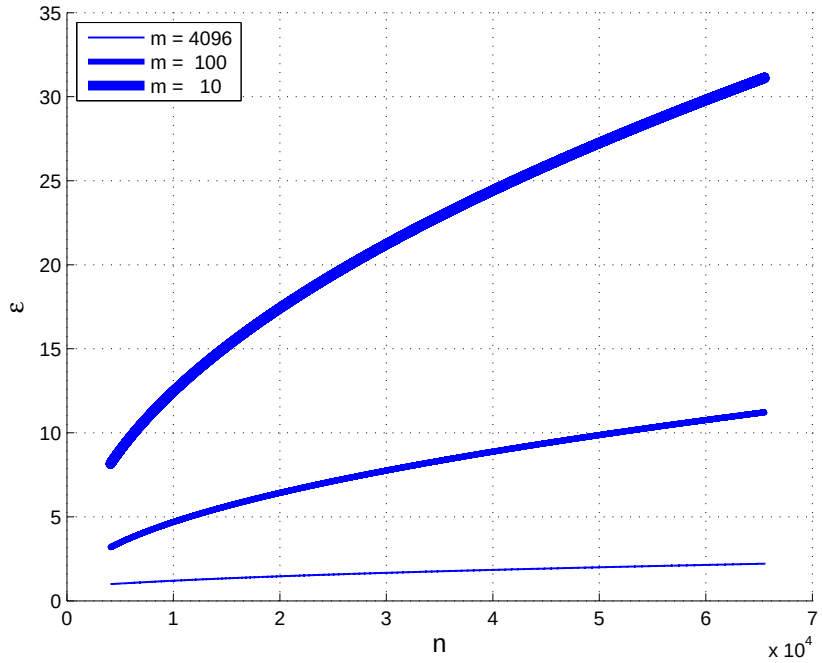

Figure 4. Average bit errors needed to corrupt a payload in an (m, n) code, when words are chosen uniformly and an adversary has full knowledge.

We assume the code is a perfect error-correcting code. According to the Hamming bound

$$\sum_{k=0}^{R} \binom{n}{k} = 2^{n-m}$$

where R is the number of errors the code can correct. And by the information theoretic inequality

$$\sum_{k=0}^{R} \binom{n}{k} \leq 2^{nH(R/n)}$$

we have

$$n - m \leq nH(R/n)$$

Here we choose

$$R = nH^{-1}(1 - \frac{m}{n})$$

Let ϵ denote the minimum necessary bit error as

$$\epsilon = R + 1 - ED$$

where ED is the expected distance from the central codeword,

$$ED = \sum_{k=0}^{R} k\binom{n}{k} / \sum_{k=0}^{R} \binom{n}{k}$$

In this case, if $\epsilon \leq n \cdot \epsilon_o$, an attacker can change the decoding by flipping ϵ bits within his attack budget. So we want to know, for a fixed m, choose $R = nH^{-1}(1 - m/n)$, as n gets large, how the minimum necessary bit error ϵ grows. The results are shown in figure 4 .

The upshot of these results is that for practical transmissions, the warden can break a code with a very small number of bits. This is regardless of the actual coding algorithm employed: no matter how clever the algorithm, the grand majority of transmissions possess one or two fragile bits.

5. TRANSMISSION UNDER AN ACTIVE WARDEN WITH PARTIAL INFORMATION

We can possibly employ a post-block approach to prevent Wendy from having full knowledge. For example, we can choose a random 128-bit secret key, send a codeword XORed with a secret key stream, and then immerse the key in the channel long after Wendy has the opportunity to damage bits. In this case, she dosen't know which bit she need flip to maximize the decoding errors. In this case, Wendy will randomly flip the bits in the stream within her noise budget ϵ. We want to know the probability that the decoding will fail. We define the probability as attack probability P_{attack}.

For the perfect error-correcting code we mentioned in the previous discussion, P_{attack} is computed as follows:

$$P_{attack} = \sum_{u=0}^{R} P(|s| = u) \cdot P(|s \oplus \eta| > R)$$

Where s is the message bit string, and η is the attack bit string. Then, we get

$$P_{attack} = \frac{1}{\sum_{k=0}^{R}\binom{n}{k}} \cdot \sum_{u=0}^{R} \binom{n}{u} \cdot \sum_{w=0}^{w<\frac{u+t-R}{2}} \frac{\binom{u}{w}\binom{n-u}{t-w}}{\binom{n}{t}}$$

Where

$$t = \lfloor n \cdot \epsilon_0 \rfloor$$

Figure 5 shows the computed probability of a transmission error for payloads of 10, 100, and 4096 bits, where n varies over 10^4 to 10^5 bits. As this shows, even under a limited adversary a single random transmission is impractical.

In practice, this result only applies to a single public transmission where transmitted bits must be genuinely uniform, rather than computationally indistinguishable from uniform. A practical key exchange is still possible if Alice and Bob are not constantly subject to attack; however, a determined adversary can in theory prevent key exchange from occurring in the first place.

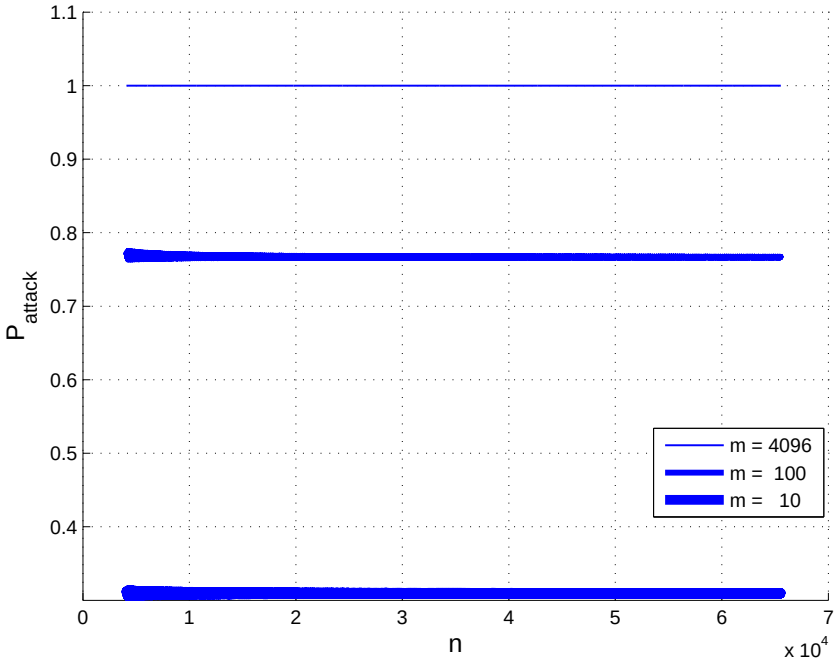

Figure 5. Error probabilities for different payloads m in an (m, n) code, when words are chosen uniformly and a warden has no extra information.

6. KEY EXCHANGE PROTOCOLS

Given a secret key, Alice and Bob can outwit an active warden with secret synchronization markers and standard error correction, while efficiently passing datagrams over the channel. How can they share that key in the first place?

We employ the technique of uniformly generating a key exchange datagram, representing it as a string of *i.i.d.* bits, and immersing it in the channel. The results from the previous section imply that a determined adversary can defeat a transmission with few errors; so rather than attempt any error correction we transmit as many blocks as possible. Synchronization is made possible by *supraliminal synchronizaton markers*: higher-level aspects of the channel that clearly denote a drop point for data. In our videoconferencing channel, supraliminal synchronization mark- ers are provided by ma jor transitions in our ciphertext-leaking computer ani- mations. In our animated raincloud backdrop, for example, PRNG state is be- trayed through the position of raindrops positioned behind the user. Occasional lighting bursts provide synchronization markers every few seconds, marking the position of potential data drops.

6.1 Choice of public key algorithm

Alice and Bob must exercise care in their choice of a public-key encryption method for key exchange, because the key exchange datagrams must have no observable structure. RSA, for example, is not allowed, because while a randomly generated RSA modulus can be immersed within a stream of coin flips, the value is observably difficult to factor.

We choose Diffie Hellmann, with a known prime p and generator g, for which the discrete logarithm problem is known to be hard. This gives us a medium for public-key cryptography that lacks any overt structure in the key exchange datagrams, and over which a datagram can be chosen uniformly. To send a key exchange datagram, Alice chooses a value h uniformly over $\{1, p-1\}$ and computes $H = g^h \pmod p$. This value is uniform over $\{1, p-1\}$; a simple procedure, described in[4] makes it uniform over the range $\{0, 2^n - 1\}$ and thus equivalent to an n-bit string of *i.i.d.* bits.

6.2 Matching and key exchange

Our overall strategy is to use the channel to continously send Diffie-Hellman exponents, and Alice and Bob repeatedly attempt to combine them to form a key. Our channel is thus divided into blocks suitable for a key datagram exchange (e.g. 4096 bits in length,) and at each block transmission an attempt is made by both parties to combine Alice's and Bob's most recent blocks. If the channel is very error-prone, each party may employ the more computationally intensive step of combining his or her most recent exponent with the last few exponents transmitted by the other.

Once a purported key is generated, Alice and Bob must determine if they have found a key. This is achieved by transmitting short hashes of the keys they have generated. Once a hash is transmitted by Alice that matches one computed by Bob, or vice versa, they now know that they have generated a secret key. Again, they could perform the slightly more computationally intensive step of computing Hamming distances: if a hash is corrupted by the warden, Alice can still spot a hash in the data stream if it is very close to one she computed.

Our key exchange protocol is then as follows:

1. For each available data block B_k, Alice transmits an encoded value $A_k = g^{a_k} \pmod p$. Likewise, at each opportunity Bob transmits an encoded value $B_k = g^{b_k} \pmod p$.
2. Upon receiving each block B_k, Bob decodes a purported A_k and computes $K_{kj} = A_k^{b_j} \pmod p$ for $j = \{k, k-1, \cdots k-c\}$. This generates $c+1$ possible keys per step; Alice computes a similar set of keys from Bob's purported B_k.
3. For each possible key K_{kj}, Alice and Bob compute a short m-bit hash, which is transmitted in a future block; for example, upon collecting enough hashes to fill as much space as a single datagram (e.g., 51 80-bit hashes to fill a 4096-bit block.) Two distinct hashes of each key are stored locally, while only one is transmitted.
4. When Alice observes a block with an aligned m-bit string that matches a line in her hash table, she transmits the second one in a future block as a confirmation.

Suppose a block has an overall probability P of correct receipt. If $c = 0$, there is a P^2 probability that a hash will match. Thus the expected number of transmissions needed for a match is

$$\sum_1^\infty (1-P^2)k(P^2)^{k-1} = \left[(1-x)\frac{d}{dx}\sum_0^\infty x^k\right]_{x=P^2} = 1/(1-(1-P^2)) = 1/P^2$$

.

For $c > 0$, we can use the same derivation but replacing the transmission probability P with a modified probability P' of some recent pair of blocks matching. Since there are $c+1$ of Alice's datagrams combined with $c+1$ of Bob's datagrams, and a single match is a success, the modified success probability is

$$P' = 1 - \left(1-P^2\right)^{(c+1)^2}$$

. For example, for a 0.5 block error rate, the expected number of block transmissions needed is 4 with $c = 0$, 2.29 for $c = 1$, and 1.73 for $c = 2$.

7. DISCUSSION AND CONCLUSION

This paper's focus is essentially post-steganographic: we imagine that an ideal robust steganographic algorithm exists, allowing Alice and Bob to cloak arbitrary messages in plausible cover data. However, even with such an advantage they cannot simply exchange a key through the channel if the Warden possesses even marginal powers to add noise.

While this problem is unavoidably discrete, a continuous analog helps to explain the problem. The set of all n-bit transmissions can be viewed as an n-dimensional space. The set of all transmissions that decode to a single

message m is an n-dimensional cell. The requirement that we transmit uniformly over the whole space requires that if we transmit m, the transmitted vector must be chosen uniformly over the cell. The warden successfully corrupts this signal by pushing it over the cell boundary, to decode to an adjoining codeword. Now, if we choose a point uniformly over an n-dimensional region in space, what is the expected distance to the boundary?

Essentially, the problem is a paradox of high dimensions: virtually all of a hypersphere's interior is right next to its boundary. If a point is chosen uniformly in an n-dimensional hypersphere of radius r, the expected distance to the boundary is $r/n + 1$. Thus for any code, if a codeword is chosen uniformly over a codeword cell then it is likely to be within a bit error of an adjoining cell. If the warden has the power to choose a worst case error vector, then she can inflict severe damage with a handful of bits unless space is divided into impractically few regions.

If the warden adds noise randomly rather than intelligently, the continuous analog is choosing a uniform point in a hypersphere of radius 1, envisioning an error sphere of fixed radius centered at that point, and asking what fraction of that error sphere's surface lies outside the original hypersphere. The attacker's success rate is the expected value of this error probability, over all signal points; but since most signal points are near the edge, most error spheres stick out of the hypersphere by a substantial margin. In short, uniformly generated data is almost always fragile.

Despite this pessimistic result, it only shows that an eternally vigilant warden can prohibit key exchange with an occasional bit error. Once a successful key exchange occurs, communication thereafter is fairly simple. We also have yet to explore the possibility of computationally secure transmission: our results are only pessimistic if we genuinely transmit uniform datagrams, rather than datagrams that are computationally indistinguishable from uniform. Whether this lower standard allows improvements in transmission is an open problem.

REFERENCES

[1] Simmons, G. J., "The history of subliminal channels," *IEEE Journal on Selected Areas in Communications* **16**, 452–462 (May 1998).

[2] Diehl, M., "Secure covert channels in multiplayer games," in [*Proceedings of the 10th ACM Workshop on Multimedia and Security*], 117–122 (2008).

[3] Hernandez-Castro, J. C., Blasco-Lopez, I., Estevez-Tapiador, J. M., and Ribagorda-Garnacho, A., "Steganography in games: a general methodology and its application to the game of go," *Computers and Security* **25**, 64–71 (February 2006).

[4] Craver, S., Li, E., Yu, J., and Atakli, I., "A supraliminal channel in a videoconferencing application," in [*Lecture Notes in Computer Science v. 5284*], 283–293 (2008).

[5] Wayner, P., [*Disappearing Cryptography*], Morgan Kauffman, San Francisco (2008).

[6] Craver, S., "On public key steganography in the presence of an active warden," in [*Lecture Notes in Computer Science v. 1525*], 355–368 (1998).

[7] Trappe, W. and Washington, L. C., [*Introduction to Cryptography with Coding Theory, 2nd edition*], Prentice Hall, Upper Saddle River, NJ USA (2005).