

# Visualizing Your Key for Secure Phone Calls and Language Independence

Michael Oehler

Dhananjay Phatak

John Krautheim

Department of Computer Science and Electrical Engineering  
University of Maryland Baltimore County, Baltimore, Maryland, USA  
{michael.oehler, phatak, john.krautheim}@umbc.edu

## ABSTRACT

We present a method to visualize and authenticate a cryptographically negotiated key for a secure phone call. That is, each caller is presented with a graphical representation of the key and through verbal interaction (i.e., side-channel authentication) they describe what they see. If they agree, the key is authenticated and the secure media session continues. The strength of the approach lies in the vocal recognition of the callers, and their ability to confirm the image displayed by their system. The necessary degree of visual recognition is achieved by using basic shapes, color and count. People, regardless of language or age, can easily identify these images. Our experience shows that they can communicate what they see with little effort and terminate the call when they differ. We believe that this approach reverses the current trend in security to divest users from the underlying cryptographic principles supporting secure systems by abstracting these principles to a comprehensible and visual form. This paper demonstrates that visualization and the human factor can play a pivotal role in establishing a secure communication channel. This short paper discusses how a key is visualized and provides some initial user feedback. We have named this approach the Short Authentication SymbolS Visually (SASSY.)

## Categories and Subject Descriptors

K.6.5 [Computing Milieux]: Security and Protection - Authentication; H.5.2 [Interfaces and Representation]: User Interfaces - Graphical User Interfaces

## General Terms

Security, Design, Human Factors

## Keywords

Visual authentication, side channel authentication, human-assisted authentication, authenticated key agreement

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VizSec '10, September 14, 2010, Ottawa, Ontario, Canada.  
Copyright 2010 ACM 978-1-4503-0013-1/10/09 ...\$10.00.

## 1. INTRODUCTION

Cyber security tasks are present at all layers of information processing, and many tasks are delegated to automated systems: applications and their cryptographic protocols, audit systems to record exceptional events, intrusion detection systems to monitor anomalous network events, and tools to provide situational awareness. Other security tasking requires cognitive recognition and involves human analysts so that appropriate response actions can be taken. Modern tools assisting these analysts are designed to amalgamate massive data sets from a variety of network sensors and provide the cyber analyst with capabilities to visually and thus, intuitively reveal the most relevant events. While profound, information security has generally neglected participation by the end user, and reasons have been assessed for some time [19]. We seem reluctant then to advance beyond passwords and simple lock icons in our browsers. It thus, seems prudent to bridge the complexity of cryptographic concepts with a visual abstraction that a user can readily understand and one that lets a user complete a cognitive task to ameliorate the security posture of the system.

We propose a visual interface to authenticate a cryptographically negotiated key for a secure phone call. This interface, which we will refer to as the Short Authentication SymbolS visually (SASSY), presents an arrangement of geometric shapes to the user, and authenticates this key through vocal recognition and acknowledgement. SASSY appears as a single *popup* requesting confirmation during call setup. On the surface, this seems to be an overly simple solution, but is in fact rooted in sound cryptographic principles. This simplicity should not be confused with its effectiveness, and ability to bridge automation, user involvement, and sound security practices. Our method is predicated on the salient features found in PGPhone and ZRTP, but provides relevant improvements. The approach in ZRTP is textual and emphasis is given to words with great phonetic distance [8]. While this certainly assures that the words can be properly interpreted, the approach is limited to English speaking users. Our method decouples this language dependency, and avails a sound security measure to broader group regardless of language or age. In general, people excel at identifying shapes, and a visual representation of the key, lets a user make a conscious security decision with ease and without having to understand cryptographic theory.

## 2. BACKGROUND

In 1996, Phil Zimmerman et al. introduced *biometric signatures*, a novel solution designed to authenticate the secret key as part of PGPfone, and the solution is currently integrated into ZRTP as the Short Authentication String (SAS) [22] [21]. The solution stems from the fact that two acquaintances can identify the other’s voice and that each trusts the other to accurately read a short list of words. Functionally, this list is derived from the negotiated Diffie-Hellman (secret) key and if the communicants hear the same words, the users are assured that only they possess the secret key. No man-in-the-middle (MITM) attack had been mounted against them.

Operationally, this solution is extremely lightweight, requires minimal computation, requires no additional bandwidth other than that used to speak the words, and does not require significant organizational support. Users read a few English words when a popup window appears at the beginning of their conversation (i.e., shortly after the secret key is negotiated) and immediately hang up if there are inconsistencies. The solution is well suited for VoIP communications requiring an expectation of privacy. Examples include tight social networks, family members, business contacts, or other situations where the callers can recognize the other’s voice. There is a chance that an intruder could mount a successful MITM attack by impersonating both callers in real-time, but the ability for a person or even an automated system to maintain this on both sides of the conversation, for long durations, and over multiple calls seems improbable at this time; this is a nominal weakness.

### 2.1 Related Work

Authentication is a tenet of information security; it is simply not possible to enable other security measures with an unknown identity. A thorough discussion on this topic would cover aspects of computer security, network security, numerous authenticated security protocols, and the endeavors by the luminaries in the field of security. We offer the following as a prelude to this topic: [5] [18] [6].

Here, we present a less common approach that uses a side channel to achieve authentication. The side channel assures that the user is in fact communicating with the intended person or device, since the side channel is not readily accessible to the adversary. The side channel thus, provides a medium to authenticate presented cryptographic parameters. These channels typically involve the user or their cognitive ability to compare and verify the parameters. Clearly, the task must also be tailored for a person. Asking them to compare long strings of hexadecimal values would be fraught with errors and met with resistance. For instance, users would read four digits of the key from a small screen on the AT&T TSD 3600. Thus, researchers devise representations that are easy to comprehend and compare.

Perrig and Song exemplify this in [16] where a root certificate is authenticated with Random Art. The art is actually a visual representation of a cryptographic hash. The user visually compares the artistic image provided by a trusted source with that generated by the host computer before accepting the certificate. In See-is-Believing, McCune photographs a barcode located on a device of interest with their camera-phone. The phone then compares the encoded value

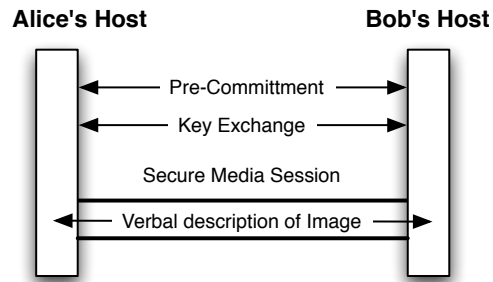


Figure 1: Key Exchange Framework

of the barcode with that of a transmitted public key (sent over an insecure link), and if they match, the key is authentic [11]. McCune also indicates that pointing at a device is a form of demonstrative identification, an intuitive and natural way for people to identify and authenticate devices around them. In [7], a system called *Loud-and-Clear* encodes the cryptographic hash of a key as a MadLib, a humorous English sentence (See [20].) This MadLib is affixed to the side of a device (e.g., a printer.) A person will then present their personal device, receive the printer’s public key, and then compare the Madlib displayed or spoken by their device with that displayed by the target device. If they match, the key is authentic. In [9], a message is encoded as a three dimensional (3-D) scene and sent to the user. The user is presumed to be on a system that may not act in the user’s best interest, but is still required to (securely) communicate with a trusted source. Here, the mal-intent computer is unable to understand the message sent from the (trusted) remote computer.

Each of these side channel examples demonstrate a different medium, visual or audio, and demonstrate the different roles used in human assisted authentication: the person assists, but the host computer decides [11], or the person compares and decides [16] and [7]. Our approach reflects the latter; the users participate in the description of the image and decide if there is a match.

## 3. SASSY

Fundamentally, a language-independent solution could require that people speak the hexadecimal digits of the hash of the key, e.g., a PGP fingerprint or a Global Unique Identifier (GUID). However, people are not well suited to speaking long strings of digits. For this reason, others have suggested human readable forms to express a key [12]. But, it seems that these forms do not provide the necessary impact to entice full interactivity with the user. People are better suited to identifying visual input.

We considered displaying landmarks, international road signs, corporate logos, and media icons, but reliance on visual cues, even apparent notable ones, may not possess the wide-ranging recognition necessary to support authentication. Instead, the necessary degree of recognition can be achieved by using basic geometric shapes, and the necessary number of combinations can be achieved through color and count. Functionally, the combinations of color, count, and shapes uniquely map to a byte value.

### 3.1 Methodology

Our method can be expressed in a basic key exchange framework and as shown in Figure 1 with three stages: A pre-commitment, the key exchange, and a verbal description of the image. Pre-commitment is designed to prevent a steering attack. That is, the pre-commitment prevents an adversary from seeing a public key before choosing theirs, and diminishes any advantage that may be gained. Namely, to prevent an adversary from opening two separate negotiations such that identical authentication images were presented. A pre-commitment is generally a cryptographic hash of the ephemeral public key and compared to a calculated hash during the exchange. The framework is independent of a particular key exchange, but it is natural to employ a traditional Diffie-Hellman exchange or Elliptic Curve Diffie-Hellman. The method then derives and presents a visual encoding from the secret key, and on each system.

Specifically, we define two sets,  $G_{even}$  and  $G_{odd}$  of glyphs so that altering glyphs are presented to the user (more on this below.) Each glyph is then defined as the concatenation of symbols from ordered-sets  $N$ ,  $C$ , and  $S_{even}$  or  $S_{odd}$ , for the count, color, and shape respectively. There are 8 elements in sets  $N$ , 4 in  $C$ , and 8 in sets  $S$ , producing 256 glyphs for each set  $G$ . See Appendix A for a realization of these sets. As an example, the first glyph in the even set will be:  $g_{even_1} = n_1|c_1|s_1$ , where  $n_1$ ,  $c_1$ , and  $s_1 \in N, C, S_{even}$  respectively. This glyph,  $g_{even_1}$  would associate to “one red square.” The rest of the set is simply formed with three iterative loops. The same structured concatenation for  $G_{odd}$  with  $S_{odd}$  is used, but not discussed further for brevity.

The glyphs from sets  $G$  are then mapped to a cryptographic hash of the secret key,  $k$ . That is, the least significant byte from the hash is used as an index into  $G_{even}$ , the next byte into  $G_{odd}$ , and so on. This mapping in aggregate forms an image for the users,  $I_A$  and  $I_B$  respectively. In Figure 2, four glyphs are depicted corresponding to 4 bytes of a hash. This type of truncation is a common practice [2]. According to [10], there are some analytical advantages of truncating the output, but the results are not absolute in this area. Our method is not limited, and more than four glyphs can be shown. The resulting image could easily be extended to the full hash width, but at the cost of burdening to the user.

The two sets of glyphs and the interleaved construction of the image assures that no two glyphs are repeated in order, and intended to facilitate the ensuing description. But, the appearance of an alternating and identical glyph is still possible. Consider the low order bytes of a contrived hash:  $0x...AAAAA00$ . The glyphs selected from the odd set,  $G_{odd}$  are the same. Although it may seem that three identi-

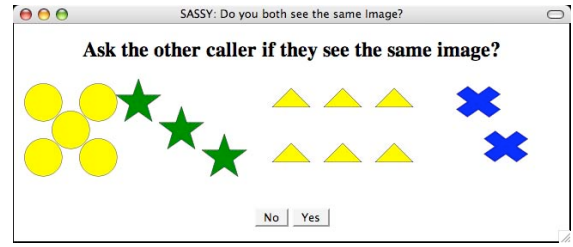


Figure 2: Example Image from SASSY

cal glyphs would be presented, the other value,  $0xAA$  selects a glyph from  $G_{even}$ , and is thus, unrelated. As a second example, consider the hash:  $0x...AA00AA00$ . Only two alternating glyphs would be seen in a four-glyph image, and would make a precise description quite challenging.

We created a linear fallback and replacement algorithm to eliminate such duplicates. This algorithm employs a second hash,  $H(H(k))$  as a source of replacement values. If two adjacent even (or odd) bytes are identical, the next unused byte from this hash is used as a replacement. If the replacement byte is a duplicate itself (an unusual case), then the next byte is employed in sequence. The algorithm scans the even bytes first, comparing the current byte to the prior even byte, and replaces the current value if they are identical. The odd bytes are scanned next. Lets assume that the replacement hash is:  $0x...332211$ . A hash value equal to  $0x...AA00AA00$  would be transformed to:  $0x...2211AA00$ . When both systems possess the same key, both will be able to construct this replacement hash. Hopefully this is evident.

When an image is created, we need to assure that the second hash can replace all duplicates that may appear in the image, even one derived from a complete hash value. This is achievable since only one byte in an identical pair is replaced, and thus, at most only half the bytes in a hash could be replaced, assuming the replacements are not duplicates. While it is possible to imagine a scenario where all replacement values are duplicates, this would be a rare event indeed. We constructed a simulation in *Mathematica* to assess this notion. The simulation generated ten thousand unique SHA-1 hash values for images, 4 to 20 glyphs in length. The simulation then counted the number of single, double, and triple replacements in a hash value. Table 1 shows that most were single replacements and there were a few double replacements. We encountered 1 triple replacement and while possible, we did not encounter any further replacements. In practice, the replacement hash should address the majority alternating and identical glyphs.

An image is then presented and if the acquaintances confirm that they both see the same image, the secure media session continues. Lastly and hopefully apparent, the visual encoding changes for each session, authenticates the negotiated key used in a protocol providing strong confidentiality, and relies on the user’s ability to confirm familiar shapes (and not something from memory like a password.)

We now present the one-pass Elliptic Curve Diffie-Hellman (ECDH) protocol with pre-commitment used to derive a key and our authentication images. A discussion of cryptography and security protocols can be found in [13], and

Table 1: Simulation of the Replacement Algorithm

Glyphs in Image	Number of Replacements:		
	Single	Double	Triple
4	57	2	0
8	242	5	0
12	382	6	0
16	499	7	0
20	637	26	1

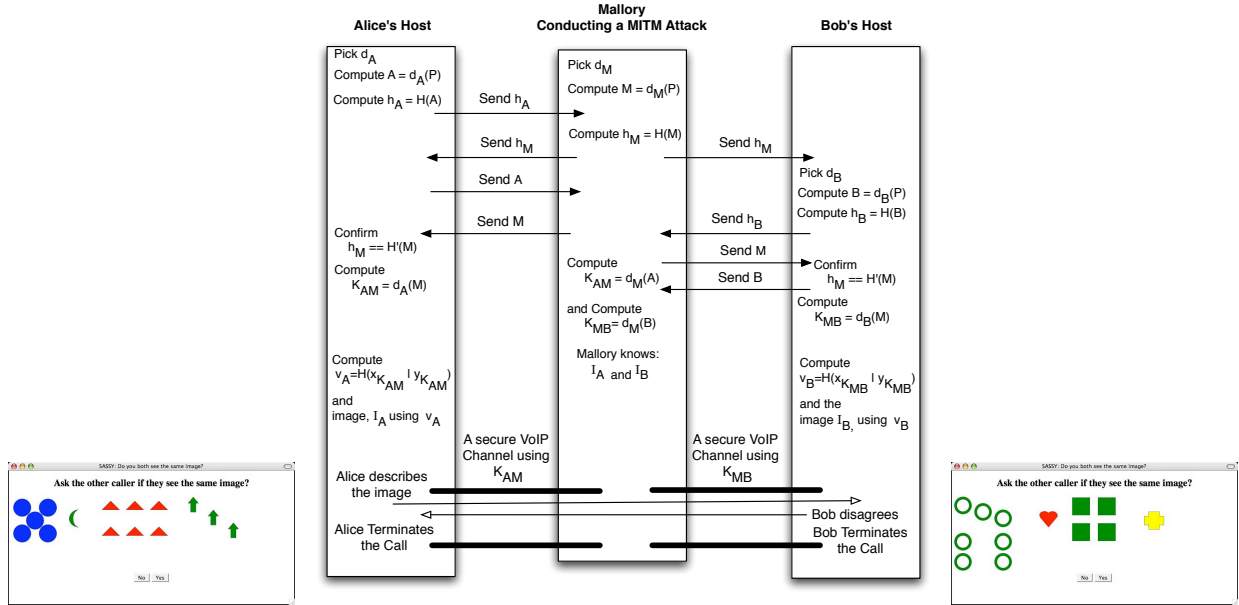


Figure 3: Example MITM Attack: SASSY Presents Different Images to Alice and Bob

discussions on parameter sizes providing comparative security levels for RSA, discrete logarithm systems, and Elliptic Curve Cryptography (ECC) can be found in [3] and [15]. A thorough discussion on the principles of ECC can be found in [4]. The benefits of ECC have been recognized for over a decade. For the same level of security as RSA, ECC offers a smaller key size, greater computational efficiency, uses less memory, and uses less channel overhead [14]. Many VoIP user-agents are computationally constrained and deployed in user environments accustomed to call setups with low latency. ECC can assure that these constraints are met when security is applied to VoIP. The choice to incorporate ECC in our approach reflects this assessment of public key cryptography. The intent is to cover the salient aspects of the approach and not to reiterate the intricacies of ECC.

Let  $D = (p, a, b, G, n, h)$  be the elliptic curve domain parameters, where  $a$  and  $b$  are the two coefficients that define the equation of the elliptic curve,  $E$ ,  $G$  is the public point,  $G = (x_G, y_G) \in E(\mathbb{F}_p)$ ,  $n$  is the order of  $G$ ,  $h$  is the cofactor, and  $p$  is the (prime) field order. Alice chooses a private key,  $d_A \in \mathbb{F}_p$  at random, computes her ephemeral public key,  $A = d_A G$ , and sends this public point,  $A$  to Bob. Bob does the same: chooses a private key,  $d_B \in \mathbb{F}_p$  at random, computes his ephemeral public key,  $B = d_B G$ , and sends  $B$  to Alice. Alice then computes the point,  $K = (x_K, y_K) = d_A B$  and Bob computes  $K = (x_K, y_K) = d_B A$ . Alice and Bob calculate the same point, since  $d_a(B) = d_B(A) \rightarrow d_A(d_B G) = d_b(d_A G)$  is identical. This secret key,  $x_K$  is then typically used as a master key or session key in a symmetric algorithm. It is assumed that the domain parameters are verified or provided from a trusted source like [1], and that public keys are validated as per expected practice for ECC.

A cryptographic hash is then applied over the byte concatenation of  $K$ , namely  $v = H(x_K || y_K)$ . The lower word (32-bits) from the hash,  $v$  is mapped to a visual image, lo-

cally created on both systems and presented to the users.

In the case of a MITM attack, the eavesdropper, Mallory will negotiate a unique secret key with Alice and Bob. Since ECDH is a key agreement protocol, where each party contributes to the creation of the secret key, Mallory cannot affect the construction of identical keys within Alice and Bob's system. Even if Mallory uses the same private key, different secrets are created and thus, different images will be presented to Alice and Bob. As seen in Figure 3, the exchange between Alice and Bob with Mallory in between, produces different keys:  $K_{AM}$  for Alice and  $K_{BM}$  for Bob. Since Alice and Bob are unable to agree that the same image is presented to them, the call must be terminated.

### 3.2 Experimentation

A proof of concept was constructed from open source components: Secure-0.4, Perl, and Firefox. Secure-0.4 is an open source toolset that implements a selection of asymmetric algorithms based on Elliptic Curve Cryptography (ECC) [17]. The Perl script was created as a test harness: The script would establish a network connection, initiate a key exchange, call the ECC executable to construct the public key, and then transmit the public components over the network. The script would then create and present a Scalable Vector Graphics (SVG) image for authentication (A representative glyph appears in Appendix B.) The script could also present random images to see if users could identify incorrect images. Firefox, a web browser was then used to automatically display the resulting SVG image. Note that this was an experiment in which people were close to one another and the images presented on a laptop.

We provide some initial insight involving six people who spoke English and four who spoke Mandarin fluently. In total then, there were five test sessions as two acquaintances were used each time, and for each test, there was an observer who could follow the conversation, whether they spoke En-

glish or Mandarin. As an initial step, we asked each person to individually describe a few glyphs (a single shape repeated a few times.) Each person spent only a brief moment on this task. None had any difficulty, but most were perplexed at the simplicity. They wondered if the task was supposed to be more complex. It was then explained that this was part of a process to assure that their conversation would in fact be private, and only if the shapes matched. With respect to the shapes, a Mandarin speaker noted that the word for heart and star were homonyms, and had to describe a star as the “5-pointed star.” We encountered the greatest variation in the description of the ring from our English speakers. Nearly each had a different interpretation, including “O”, tire, and doughnut. None of these variations were a hindrance in practice.

The script was then activated and the acquaintances were presented a number of images to verify, and by the end of the series, most people started shortening their description. They would neglect a color or count. For example, “three red circles” would be truncated to “red circles” or “three circles.” At first this may seem to be a severe deficiency. However, when different images were presented, people were able to detect the ruse nearly immediately even if they abbreviated the description. Upon hearing the description of the first and non-matching glyph, the listener would interject with a response to counter the description. A rapid dialogue generally ensued fully describing the images. It was evident that they were not presented the same image. Both would agree that they saw different images and terminate the call. This truncation provides an opportunity for duality in the conversation, each has a chance to speak in the protocol exchange. It seems that when a person neglects certain details and the other provides them as part of the conversation, that this only reinforces the requirement that there be vocal recognition of the callers.

## 4. CONCLUSION

We devised a method to visualize and authenticate a cryptographic key for a secure phone call that is independent of the caller’s language. Our approach is extremely lightweight, requires minimal computation, and requires no additional bandwidth other than that used to describe the image. The approach intuitively involves the user and shows that people can play a pivotal role in the security process. The strength of SASSY is founded on strong cryptographic principles and the caller’s ability to confirm an image based on this cryptographic exchange. Our experience has shown that people are able to quickly describe and decide whether both see the same image. Our initial findings are encouraging and are confident that a future and long-term study will formally validate these findings. We also recognize that the special needs of certain users should be addressed, as some people may not be able to distinguish the color of each shape. A variant scheme could replace the colors with patterns. Vertical, horizontal, a diagonal hash-marks, and a textured fill pattern could be employed and achieve the same objective. For those who are hard of hearing, a secure video session and sign language could be used to communicate the image. We hope our method is integrated into a VoIP client and engages the user as part of the cyber security process.

## 5. REFERENCES

- [1] SEC2: recommended elliptic curve domain parameters. Technical report, Certicom, 2000.
- [2] Secure hash standard (SHS). Technical Report Draft Federal Information Processing Standards Publication (FIPS) 180-3, NIST, June 2007.
- [3] E. Barker, D. Johnson, and M. Smid. Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised). Technical Report NIST Special Publication 800-56A, NIST, March 2007.
- [4] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. London Mathematical Society, Lecture Note Series, London, 1999.
- [5] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
- [6] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.
- [7] M. T. Goodrich, J. S. Sirivianos, G. Tsudik, and E. Uzun. Loud and clear: Human-verifiable authentication based on audio. In *ICDCS*, 2005.
- [8] P. Juola. Isolated-word confusion metrics and the PGPfone alphabet. In *New Methods in Language Processing 2 (NeMLaP-96)*, Ankara, Turkey, 1996.
- [9] J. King and A. dos Santos. A user-friendly approach to human authentication of messages. In *FC05, the Financial Cryptography and Data Security*, Roseau, The Commonwealth Of Dominica, 2005.
- [10] H. Krawczyk, M. Bellare, and R. Canetti. RFC-2104: HMAC, keyed-hashing for message authentication. Technical report, Internet Engineering Task Force, February 1997.
- [11] J. McCune, A. Perrig, and M. Reiter. Seeing is believing: Using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, pages 110–124, 2005.
- [12] D. McDonald. RFC-1751: Convention for human-readable 128-bit keys. Technical report, Internet Engineering Task Force, December 1994.
- [13] A. Menezes, P. C. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [14] National Security Agency. The case for elliptic curve cryptography, 2006.
- [15] H. Orman and P. Hoffman. RFC-3766: Determining strengths for public keys used for exchanging symmetric keys. Technical report, Internet Engineering Task Force, April 2004.
- [16] A. Perrig and D. Song. Hash visualization: A new technique to improve real world security. In *International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC’99)*, pages 131–138, 1999.
- [17] B. Poettering. Secure. <http://point-at-infinity.org/seccure>, 2009.
- [18] R. Rivest and A. Shamir. How to expose an eavesdropper. *Communications of the ACM*, 27(4):393–395, 1984.
- [19] A. Whitten and J. D. Tygar. Why Johnny can’t encrypt - a usability evaluation of PGP 5.0. In *the 8th USENIX Security Conference*, Washington, DC, 1999.
- [20] Wikipedia. Mad libs. [http://en.wikipedia.org/wiki/Mad\\_Libs](http://en.wikipedia.org/wiki/Mad_Libs), 2010.
- [21] P. R. Zimmermann, A. Johnson, and J. Callas. ZRTP: Media path key agreement for unicast secure RTP. draft-zimmermann-avt-zrtp-17, Internet Engineering Task Force, April 2010.
- [22] P. R. Zimmermann, W. Price, C. Hall, C. Plumb, J. Sorensen, W. Kinney, K. MacInnis, and P. Juola. The PGPfone owner’s manual, version 1.0 beta 7. Technical report, Pretty Good Privacy Inc., 1996.

# APPENDIX

## A. GLYPH FEATURES

The tables below represent the sets of shapes employed by SASSY, the various colors for each of the shapes, and present the count (arrangement) for each shape. In total, two sets of 256 glyphs are created from these count, color, and shapes. Note that the four color squares and the arrangement of black circles for count are representative. Any shape can be one of the four colors, and any shape can be presented one to eight times.

Table 2: Shapes: Even Set,  $S_{even}$









Square 	Triangle 	Octagon 	Circle 
Pentagon 	Diamond 	Oval 	Ring 

Table 3: Shapes: Odd Set,  $S_{odd}$

















Star 	Cross 	Up Arrow 	Crescent 
Heart 	Smiley 	Club 	X-mark 

Table 4: Colors: Set,  $C$

Red 	Green 	Blue 	Yellow 
---	---	--	--

Table 5: Count: Set,  $N$

One 	Two 	Three 	Four 
Five 	Six 	Seven 	Eight 

## B. SVG GENERATION FOR “TWO RED SQUARES”

Images are dynamically generated as a Scalable Vector Graphic (SVG) reducing storage costs. Here is how to create “two red squares”:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="600px" height="400px" xmlns="http://www.w3.org/2000/svg" version="1.1">
<desc>two Red Square</desc>
<g transform="translate(3,3)">
<g transform="translate(16,8)">
<svg width="56px" height="56px" version="1.1" viewBox="0 0 256 256" preserveAspectRatio="none">
<rect x="10" y="10" width="200" height="200" fill="Red" stroke="black" stroke-width="2" />
</svg>
</g>
<g transform="translate(48,60)">
<svg width="56px" height="56px" version="1.1" viewBox="0 0 256 256" preserveAspectRatio="none" >
<rect x="10" y="10" width="200" height="200" fill="Red" stroke="black" stroke-width="2" />
</svg>
</g>
</g>
</svg>
```