Multichannel Security Protocols

The authors' security protocols exploit additional transmissions over lower-capacity channels, typically found in ubicomp environments, that offer a different combination of security properties.

Ford Long Wong and Frank Stajano
University of Cambridge

Multichannel security protocols transmit messages over multiple communication channels, taking into account each channel's security properties. Our first intentional use of these protocols goes back to a 1999 article that proposed physical contact for imprinting as opposed to the wireless channel used in subsequent operations.[1] Only later did we understand three key points.[2] First, explicit use of multiple channels in the same protocol can offer significant advantages for both security and usability. Second, explicitly stating the properties of the channel on which each protocol message is transmitted is useful for understanding one's own protocol in greater depth and therefore for addressing subtle vulnerabilities early on. Third, multichannel protocols existed long before we recognized them as such—think of the courier handcuffed to the briefcase carrying the code book that will later protect postal or telegraphic traffic.

As protocol designers, we have much to gain by adopting the multichannel viewpoint: it forces us to be more precise about security requirements and the attacker model. Such precision of expression (and, by implication, of purpose) helps us anticipate and avoid design flaws.

Multichannel protocols are particularly relevant for ubiquitous computing, which typically involves heterogeneous communication channels—as opposed to, for example, the comparatively more uniform scenario offered by the packet-switched Internet. One of Mark Weiser's seminal articles about ubicomp asked, "Can the device communicate simultaneously along multiple channels?"[3] And if it can, we add, what are the advantages for security?

In this article, we describe several protocols that we've designed using the multichannel approach and the benefits gained in security and usability. Our work's original contribution is as much in the protocols as in the explicit adoption of the multichannel viewpoint. (The sidebar "Related Work in Multichannel Security Protocols" provides further information about the field.)

Bootstrapping ubicomp security

One classical ubicomp problem is that of forming a *security association* (a shared secret) between two devices that can talk over an insecure channel—for example, a camera phone and a shared printer linked by radio—without an authentication infrastructure.

A solution based only on symmetric cryptography lets a passive eavesdropper derive the secret. To prevent eavesdropping, the devices can form a shared secret through a Diffie-Hellman (DH) exchange.[4]

However, a *Dolev-Yao attacker*[5] (which can intercept, stop, modify, and insert messages at will) could still mount a man-in-the-middle (MITM) attack. Such a bugging device—let's call it Mallory—intercepts all messages between camera phone Alice and printer Bob, establishes an Alice-Mallory key and a Mallory-Bob key, and then decrypts, reads, and re-encrypts all messages

Related Work in Multichannel Security Protocols

Several authors have recently published significant work using auxiliary channels, including

- Dirk Balfanz and his colleagues, who use a "location-limited" channel to commit to a public key's hash;[1]
- Jaap-Henk Hoepman, who studies the ephemeral key exchange (φKE) problem with explicit consideration of various channels' security-related properties;[2]
- Jonathan McCune, Adrian Perrig, and Michael Reiter, who study the visual channel using camera phones;[3]
- Serge Vaudenay, who describes commitment schemes and introduces stronger authenticity properties;[4]
- Mario Čagalj, Srdjan Čapkun, and Jean-Pierre Hubaux, who propose three protocols based on visual and verbal interaction between the participants;[5] and
- Sven Laur and Kaisa Nyberg, with their round-efficient Mana IV protocol.[6]

Before all these, Lars Erik Holmquist and his colleagues proposed an imaginative auxiliary channel for pairing—shaking devices together;[7] the associated security protocol came later.

In terms of security proofs, Ueli Maurer and Pierre Schmid developed early on a calculus of channel security properties and transformations between them.[8] Hoepman,[2] Vaudenay,[4] and Laur and Nyberg[6] provide security proofs for their proposals. Sadie Creese and her colleagues argue that, in formalizing the attacker model in a ubicomp environment, assuming a Dolev-Yao attacker across all channels isn't necessary.[9] They consider how to model this in formal tools such as the CSP (Communicating Sequential Processes) language, the FDR (Failures-Divergences Refinement) model checker, and the Casper compiler.

Finally, industry associations such as the Bluetooth Special Interest Group, the Wi-Fi Alliance, and Wireless USB have also introduced multichannel pairing protocols in their latest proposals.

REFERENCES

1. D. Balfanz et al., "Talking to Strangers: Authentication in Ad Hoc Wireless Networks," *Proc. Network and Distributed System Security Symp.*, Internet Soc., 2002.
2. J.-H. Hoepman, "The Ephemeral Pairing Problem," *Proc. Financial Cryptography,* LNCS 3110, Springer, 2004, pp. 212–226.
3. J.M. McCune, A. Perrig, and M.K. Reiter, *Seeing Is Believing: Using Camera Phones for Human-Verifiable Authentication*, tech. report CMU-CS-04-174, Computer Science Dept., Carnegie Mellon Univ., 2004.
4. S. Vaudenay, "Secure Communications over Insecure Channels Based on Short Authenticated Strings," *Proc. 25th Ann. Int'l Cryptology Conf.* (Crypto 05), LNCS 3621, Springer, 2005, pp. 309–326.
5. M. Čagalj, S. Čapkun, and J.P. Hubaux, "Key Agreement in Peer-to-Peer Wireless Networks," *Proc. IEEE*, vol. 94, no. 2, 2006, pp. 467–478.
6. S. Laur and K. Nyberg, "Efficient Mutual Data Authentication Using Manually Authenticated Strings," *Proc. 5th Int'l Conf. Cryptology and Network Security* (CANS 06), LNCS 4301, Springer, 2006, pp. 90–107.
7. L.E. Holmquist et al., "Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts," *Proc. 3rd Int'l Conf. Ubiquitous Computing* (Ubicomp 01), LNCS 2201, Springer, 2001, pp. 116–122.
8. U. Maurer and P. Schmid, "A Calculus for Secure Channel Establishment in Open Networks," *Proc. 3rd European Symp. Research in Computer Security* (Esorics 94), LNCS 875, Springer, 1994, pp. 173–192.
9. S. Creese et al., "The Attacker in Ubiquitous Computing Environments: Formalising the Threat Model," *Proc. 1st Int'l Workshop Formal Aspects in Security and Trust* (FAST 03), tech. report, Inst. of Informatics and Telematics–Italian Nat'l Research Council, 2003, pp. 83–97.

as they come through. So there are two related problems: first, Mallory can read (and rewrite) the traffic between Alice and Bob; second, and more fundamentally, Alice and Bob believe they've established a secure channel with somebody, but they're not sure whether that is the right party.

With channels such as radio, you can't be too sure where messages come from. On the other hand, if a device you recognize as the source displays a value on its LCD, you can be sure that the value comes from that device and not from an MITM one. We describe this channel property as *data origin authenticity*. It's also practically impossible for a MITM attacker to make you see a different value on the genuine display; we refer to this as *integrity* for this channel. However, given that the screen might be visible to nonparticipants, that channel doesn't offer confidentiality. In what follows, we'll assume that a Dolev-Yao attacker operates on the radio channel but that other, lower-capacity channels are also available on which the attackers' powers are reduced—for example, just to eavesdropping.

As an example, let's see how Alice and Bob can verify, using multiple channels, whether they're both sharing the same DH key g^{ab} (we omit all the "(mod n)" for brevity). Figure 1a presents protocol 1a. The key will be the same essentially only if there's no MITM. Alice computes the key's hash, $h((g^b)^a)$, and Bob computes his, $h((g^a)^b)$. By comparing these independently computed hashes over a different channel offering data origin authenticity—for example, by displaying them on their respective LCD panels—the users can verify whether the

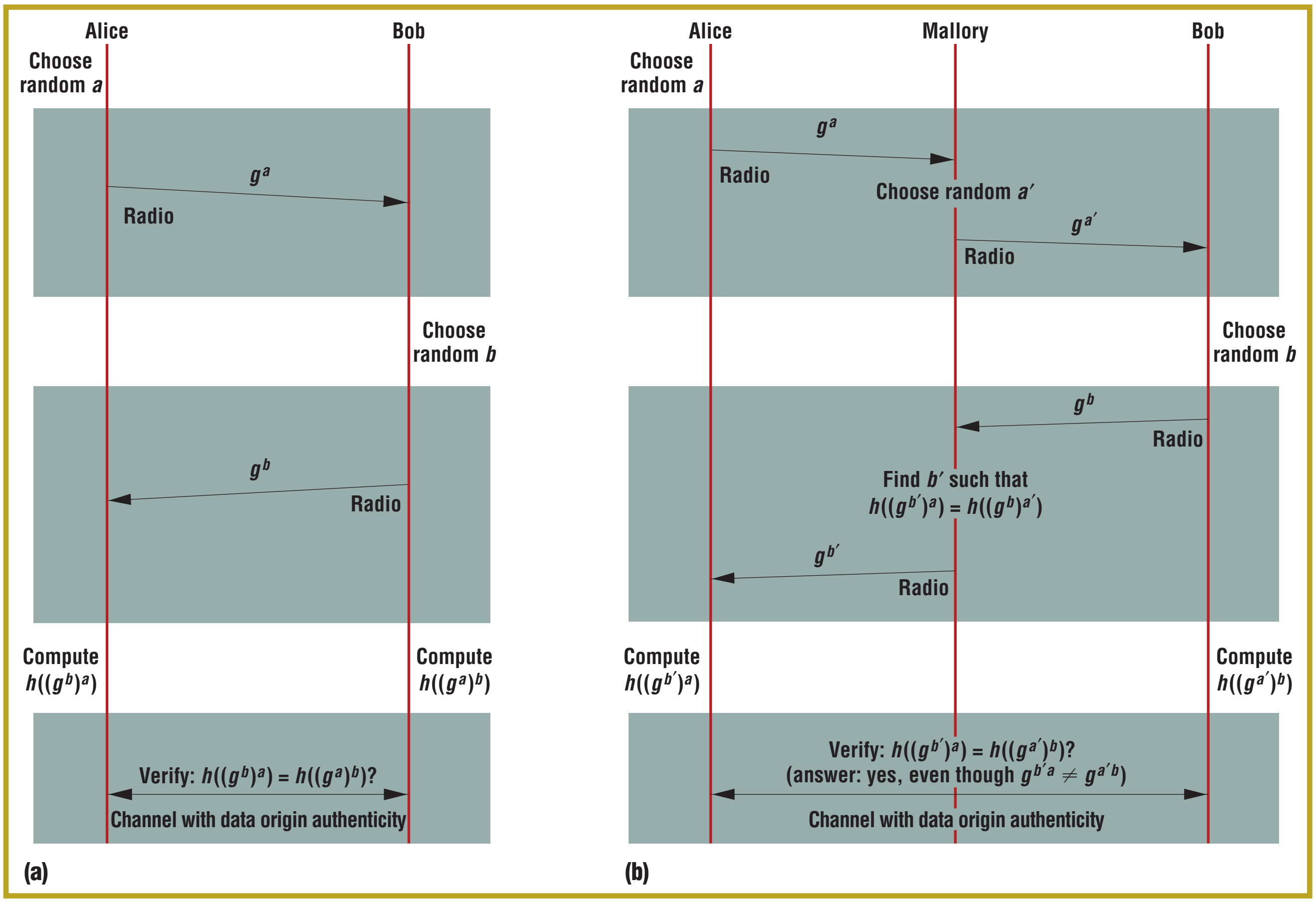


Figure 1. A Diffie-Hellman key exchange protocol (a) over a radio with hash verification on a separate channel, and (b) in the presence of a man-in-the-middle attacker.

devices share the same key. If the hashes don't match, the user can abort the transaction and call the bug-sweeping team to look for Mallory.

The figure's shaded rectangles show that protocols 1a and 1b would look essentially the same to Alice and Bob. In other words, neither Alice nor Bob could tell, simply on the basis of the messages they receive, whether there's an MITM attacker Mallory.

There are many possible ways to perform the verification over a nonradio channel. The double-headed arrow at the end of Protocol 1a is actually a subprotocol in itself that we could expand in several ways, including the following:

1. Both devices display their hash, then the user compares them and presses OK or Cancel on each device.
2. The user prints out the hash using the printer Bob and types it on the camera phone Alice's numeric keypad.
3. Using the camera phone Alice, the user takes a picture of the hash that Bob printed.

Further alternatives are possible. Assurance about the message's origin comes not only from using a visual channel instead of radio but also from actually involving the human user in the process.

Unfortunately, almost all variations of this auxiliary authentic channel have some capacity limitation imposed by usability and robustness constraints. For example, the user won't enjoy comparing or typing a hash much longer than half a dozen characters. So, using truncated hashes—say, 20 to 30 bits—might be necessary. The design of the protocols that use such short values is critical, because unsuitable design can give rise to feasible attacks.

Going back to protocol 1a, imagine that the hashes are short (that is, truncated) enough to be brute-forced in real time. Then Mallory could still mount a MITM attack despite the visual check. As protocol 1b shows, Mallory chooses the forged DH contributions via brute-force search to ensure that, when combined with the victims' own contributions, Alice and Bob respectively form keys $g^{ab'}$ and $g^{a'b'}$ that yield the same truncated hash. Consequently, the user will confirm that the truncated hashes from the two devices match and mistakenly assume that all is fine, even though the calculated session keys are actually different and an MITM attack is taking place.

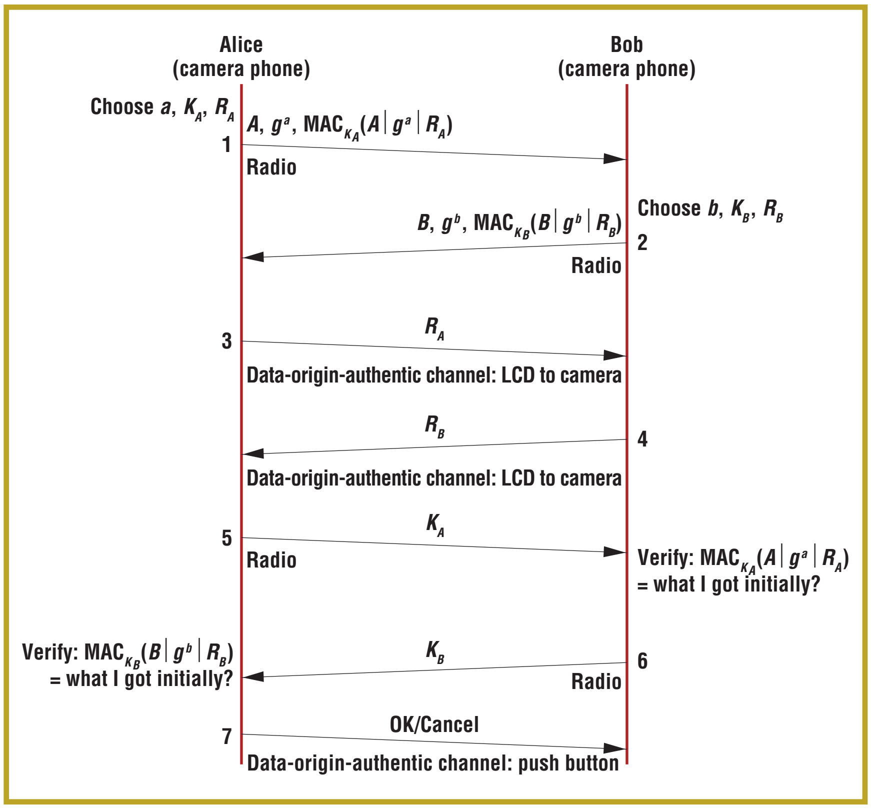

Alice (camera phone)
Bob (camera phone)
Choose a, K_A, R_A
1
A, g^a, $\mathrm{MAC}_{K_A}(A|g^a|R_A)$
Radio
Choose b, K_B, R_B
B, g^b, $\mathrm{MAC}_{K_B}(B|g^b|R_B)$
2
Radio
3
R_A
Data-origin-authentic channel: LCD to camera
R_B
4
Data-origin-authentic channel: LCD to camera
5
K_A
Radio
Verify: $\mathrm{MAC}_{K_A}(A|g^a|R_A)$ = what I got initially?
Verify: $\mathrm{MAC}_{K_B}(B|g^b|R_B)$ = what I got initially?
K_B
6
Radio
7
OK/Cancel
Data-origin-authentic channel: push button

Figure 2. Man-in-the-middle-resistant mutual authentication.

Low-capacity channels

Usability prevents us from adopting full-size hashes (greater than 200 bits) on the data-origin-authentic channel, but a robust protocol is still possible, even with truncated hashes. We must force Mallory to guess correctly in just one attempt, instead of allowing a brute-force search. Used this way, a 20-bit hash gives us a 2^{-20} (1 in a million) probability that the MITM attack will succeed (strong security), whereas previously it gave us only the modest protection of a 2^{20} workload for the brute-force attacker (weak security).

Protocol 2 (see figure 2) is a slightly more efficient version of our 2005 proposal.[2] It achieves strong security by combining long values sent over radio with short values sent over the data-origin-authentic auxiliary channel.

Imagine two camera phones (again named Alice and Bob) taking pictures of each other's screen in turn. The protocol is essentially symmetric, so let's consider Alice's side and the messages she sends. Initially, Alice randomly selects a (long) private value a, a short (for example, 20-bit) *nonce* R_A (an arbitrary number used only once per security session), and a long (for example, 256-bit) message authentication code key K_A. Alice then computes her ephemeral DH public key g^a and a commitment $\mathrm{MAC}_{K_A}\left(A|g^a|R_A\right)$. In message 1, Alice sends her identifier, the public key, and the commitment. Because this happens over radio, length isn't a problem, so the MAC itself can be long, too. In message 3, after having received Bob's public key and commitment, she releases her short nonce R_A over the visual channel. In message 5, after receiving Bob's nonce R_B, she releases her (long) MAC key K_A via radio.

At that point Bob can compute Alice's commitment using Alice's identifier and public key from message 1, her nonce from message 3, and her MAC key from message 5, and verify whether he obtains the same MAC value as the one he received in message 1. If the values match, he's assured that the g^a he received is the one that Alice transmitted, so he can safely use it to calculate the shared secret g^{ab}.

Why can Bob be assured? Consider Mallory's perspective. To substitute his own $g^{a'}$ on the way from Alice to Bob, Mallory would need to send a commitment in message 1. He can substitute a MAC key but can't anticipate (or forge) the R_A nonce that Alice will release in message 3—indeed, as we mentioned earlier, he only has, say, a one-in-a-million chance of predicting correctly. If Mallory guesses incorrectly, Bob will notice it when he receives message 5 and tries to verify the MAC.

If the protocol used short values throughout, Mallory might alternatively consider sending a random commitment to Bob in message 1 and brute-forcing a suitable MAC key after having seen Alice's nonce R_A. Mallory would search for a MAC key x such that $\mathrm{MAC}_x\left(A|g^{a'}|R_A\right)$ evaluated to the random commitment he sent Bob in message 1. If the MAC value were only t bits long, then he'd try only about 2^t keys before finding a suitable x. But this won't work in our protocol because the MAC key and output, both sent over radio, can afford to be hundreds of bits long and are therefore out of reach of a practical brute-force search.

One point to note is whether the protocol designer using an auxiliary channel assumes it to be confidential. Some previous multichannel protocols have assumed their auxiliary channels' confidentiality, in addition to authenticity. However, that assumption's validity should be reevaluated, especially for visual channels, now that surveillance cameras and camera phones are so pervasive. Our protocol, instead, does not assume confidentiality and resists eavesdropping on the auxiliary channel.

As a general principle, then, the protocol's DH components help resist eavesdropping attacks, while short values transferred over the auxiliary channel, secured by nonmalleable commitments,[6] help resist active attacks.

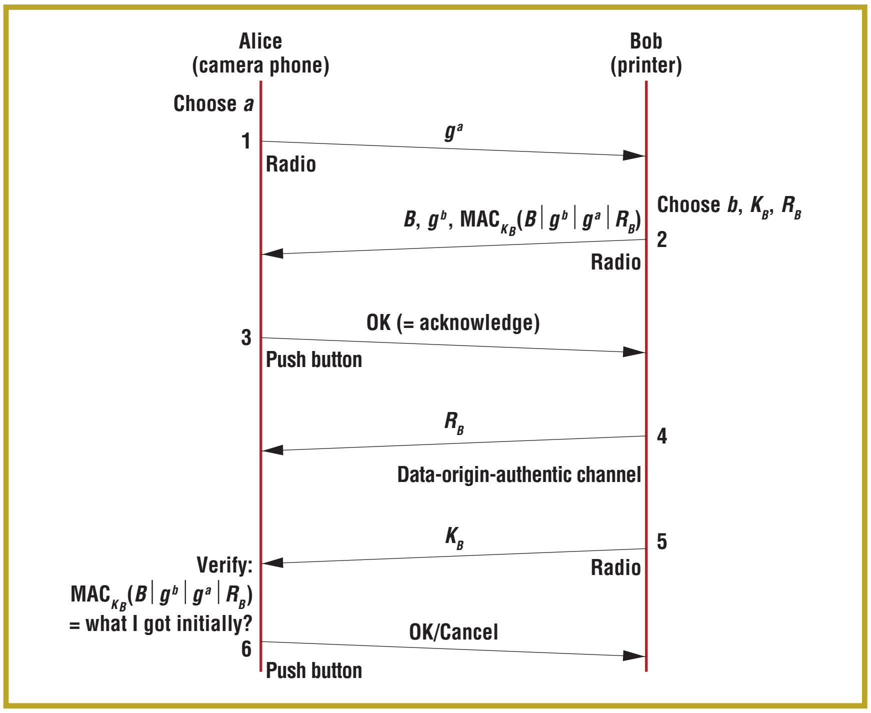


Figure 3. Strong security despite the unidirectional visual channel.

Unidirectional authentic channel

While protocol 2 must transmit short nonces in both directions, we can actually achieve the equivalent of strong Dolev-Yao-resistant mutual authentication with just a unidirectional low-bandwidth authentic channel.

Going back to imagining Alice as a camera phone and Bob as a printer, Alice can "see" Bob's short code (either on an LCD panel or printed on paper), but not vice versa. In our protocol 3 (see figure 3),[2] after receiving Alice's ephemeral public key g^a over radio, Bob responds with his own identifier, public key, and a commitment $\mathrm{MAC}_{K_B}\left(B\middle|g^b\middle|g^a\middle|R_B\right)$, where R_B is a short random nonce. Bob releases this random nonce over the authentic (visual) channel and the MAC key K_B over the Dolev-Yao-affected (radio) channel, but only after Alice has acknowledged (with message 3, which carries only one bit but which Mallory can't fake) that she received the commitment. This prevents Mallory from blocking Bob's message 2, intercepting R_B and K_B from messages 4 and 5, and then issuing to Alice a forged message 2 constructed with knowledge of R_B and K_B.

Message 3 requires only a one-bit-per-message data-origin-authentic channel from Alice to Bob, such as a pair of OK and Cancel push buttons or a single push button with a time-out (notice that pressing buttons implies a human operator). Nevertheless, message 3 forces Mallory to commit early to a guess if he wants to issue a fake message 2 to Alice.

After receiving message 5, Alice computes $\mathrm{MAC}_{K_B}\left(B\middle|g^b\middle|g^a\middle|R_B\right)$ and verifies whether it matches what she received in message 2. If the values match, she's assured that she received Bob's original g^b and that Bob received her original g^a, meaning that they can both safely calculate their shared secret g^{ab}. But Bob doesn't know yet whether the verification succeeded. So Alice must notify him, again using the one-bit-per-message authentic channel.

This protocol is remarkable because it achieves strong security even without a low-capacity authentic channel over which to transfer a short nonce from Alice to Bob. The critical insight is that Alice (assuming she's honest and uncompromised) can perform the verification and then securely notify Bob of the outcome with the human user's help. This notification's authenticity is assured by the fact that the Dolev-Yao attacker on the radio can't press the button on device Bob. With this multichannel protocol, we've essentially amplified the data origin authenticity of the one-bit-per-message push-button channel to cover both parties' DH keys g^a and g^b.

The sidebar "Multichannel Group Key Agreement" discusses how to extend such a protocol to more than two parties.

Implementation issues

To validate our protocols' feasibility, we implemented prototypes of the key building blocks (though not full implementations) and experimented with two auxiliary channels. Here are the more significant quantitative aspects.

Computing time

The DH protocol is often deemed unsuitable for ubicomp devices because of the computational load of modular arithmetic. While this is true for simpler devices such as wireless headphones and RFID tags, it's less of a concern for modern phones, cameras, MP3 players, or PDAs. We ported the relevant C routines from Shamus Software's MIRACL cryptographic library (www.shamus.ie) to the Symbian v7.0s operating system used in several Nokia camera phones. We then compiled them using the development environment of Symbian's Series 60 Developer Platform 2.0.

We obtained reasonably fast per-exponentiation timings: for elliptic-curve group key sizes of 160, 192, and 224 bits, we respectively measured 81, 118, and 160 milliseconds on a Nokia 6600 (104-MHz ARM CPU) and 68, 98, and 137 ms on a Nokia 6670 (123 MHz, see figure 4a). We obtained each measurement by averaging 1,000 exponentiations. Compared to exponentiations, MAC computations take negligible time.

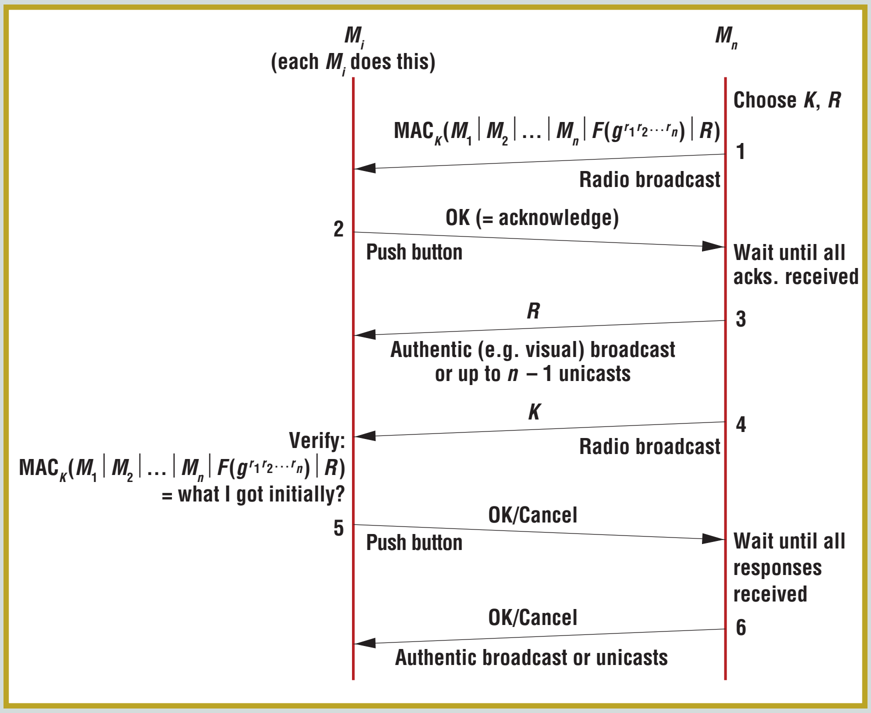
Multichannel Group Key Agreement

When more than two devices must establish a common secret, we can build on a multiparty extension to Diffie-Hellman,[1] such as the Cliques Group Key Agreement (GKA) protocol suite. The basic scheme[2] arranges the members in a linear chain: each member M_i receives a bit string from its predecessor M_{i-1} and passes it on to its successor M_{i+1} after merging its own contribution r_i to the key. The last member in the chain (M_n) broadcasts the final keying material to all other members, enabling each to compute the common group key $g^{r_1 r_2 \cdots r_n}$.

Olivier Pereira and Jean-Jacques Quisquater have shown[3] that, in the presence of a Dolev-Yao attacker,[4] the protocols in this family fail to provide implicit key authentication, perfect forward secrecy, and resistance to known key attacks. We can address these problems in various ways, but ultimately we must ensure that everyone computes the same group key $g^{r_1 r_2 \cdots r_n}$. Previously we proposed a protocol that authenticated intermediate keys' origin at every round, using commitments and auxiliary channels.[5]

A more efficient scheme, as shown in figure A, runs after an unauthenticated GKA finishes, enabling all participants to check whether they've computed the same group key. This scheme resists Dolev-Yao attacks.

Topologically, after using a linear chain for the GKA, we now have a "star" structure for the verification in which the various M_i (for $I \in \{1, 2, \ldots, n-1\}$) talk back only to the group leader M_n and not to each other, while M_n broadcasts to all others. For radio, a broadcast, instead of unicasts, is what physically happens. We make the reasonable assumption that the group leader M_n is honest, since a leader is typically chosen carefully. For the low-



M_i (each M_i does this)
M_n
Choose K, R
$\mathrm{MAC}_K(M_1 | M_2 | \ldots | M_n | F(g^{r_1 r_2 \cdots r_n}) | R)$
1
Radio broadcast
2
OK (= acknowledge)
Push button
Wait until all acks. received
R
3
Authentic (e.g. visual) broadcast or up to $n-1$ unicasts
K
4
Radio broadcast
Verify: $\mathrm{MAC}_K(M_1 | M_2 | \ldots | M_n | F(g^{r_1 r_2 \cdots r_n}) | R)$ = what I got initially?
5
OK/Cancel
Push button
Wait until all responses received
OK/Cancel
6
Authentic broadcast or unicasts

Figure A. Multichannel man-in-the-middle-resistant validation of a computed group key.

On PDAs, laptops, and PCs, computations are, of course, even less of an issue. The cryptographic code was 17,175 lines of C and, once compiled, occupied just 122 Kbytes—insignificant compared to the phones' 6 and 8 Mbytes of shared memory (further expandable with MultiMediaCards). The radio messaging is straightforward to implement using the Symbian Platform Bluetooth API.

Visual channel

2D visual-code software is now widely available. We experimented with our lab's Target Recognition Using Image Processing (TRIP) system and the Semacode's open-source software development kit on both PCs and our two camera phones. TRIP was designed to enable automatic recognition (and estimation of location and orientation) of its circular targets in a video frame of a cluttered scene. The square Data Matrix (ISO/IEC 16022) targets used by Semacode were meant to be acquired by explicitly photographing the tag straight on. So, for a given pixel count of the acquired target, Semacode tags carry many more bits and are better suited to our security application; we want the user to perform an explicit acquisition operation instead of letting the software grab any code it locks onto.

We set out to measure the maximum capacity of the phone-to-phone visual channel by encoding progressively longer strings as Semacode tags of increasing pixel count, displaying them on the Nokia 6600, acquiring them with the Nokia 6670 (the one with the better camera), and recording the largest size at which they could be transferred reliably. We spent 5 seconds attempting to acquire each frame, repeating 10 times for each frame, and we capacity authentic channel (for example, when M_i takes a picture of M_n's screen), broadcasting might be possible (for example, by connecting M_n to a projector). If not, iterating a point-to-point transmission to each M_i would also work. For the one-bit-per-message authentic channel in the opposite direction (for example, when the human operator presses the OK or Cancel button on M_n depending on M_i's result), it will be necessary to repeat for each member the point-to-point procedure and track which M_i is causing this button press.

First, M_n broadcasts a keyed commitment

$$\mathrm{MAC}_K\left(M_1 | M_2 | \ldots | M_n | F\left(G^{r_1 r_2 \ldots r_n}\right) | R\right)$$

to all M_i, using the high-capacity (for example, radio) channel, on which the Dolev-Yao attacker is assumed to operate. Here $F()$ is a pseudorandom key derivation function, R is a short random nonce (an arbitrary number used only once per security session), and K is a long random MAC key. Then, after all the M_i devices acknowledge receiving the broadcast (using the authentic one-bit-per-message channel), M_n releases the nonce R and the key K to all M_i over the visual and radio channel, respectively. Each M_i then recomputes the commitment, verifies whether it matches the one received from M_n, and reports the answer to M_n over the one-bit authentic channel. Finally, after receiving everyone's reports, M_n tells everyone over an authentic channel whether they all reported success. The channel needs to carry only one bit but could be the visual channel, because it can broadcast instead of using the push button. If even a single verification fails, the protocol will abort.

Several comparable protocols appear in the literature, such as those by Long Nguyen and Bill Roscoe[6] and Jukka Valkonen and his colleagues.[7] Our use of the authentic one-bit-per-message channel makes our protocol more efficient in some respects; we believe this illustrates the usefulness of making channel modeling more explicit in protocol design.

In message 3, if we use $n - 1$ unicasts instead of one broadcast, the unicasts don't necessarily need to take place over the same type of authentic channel. If the M_i devices are heterogeneous, they may each use their preferred auxiliary channel (camera, keypad, audio, near-field, contact, and so on). We just require that each auxiliary channel offer data origin authenticity and sufficient capacity to transmit R, and that M_n can act as a source for it.

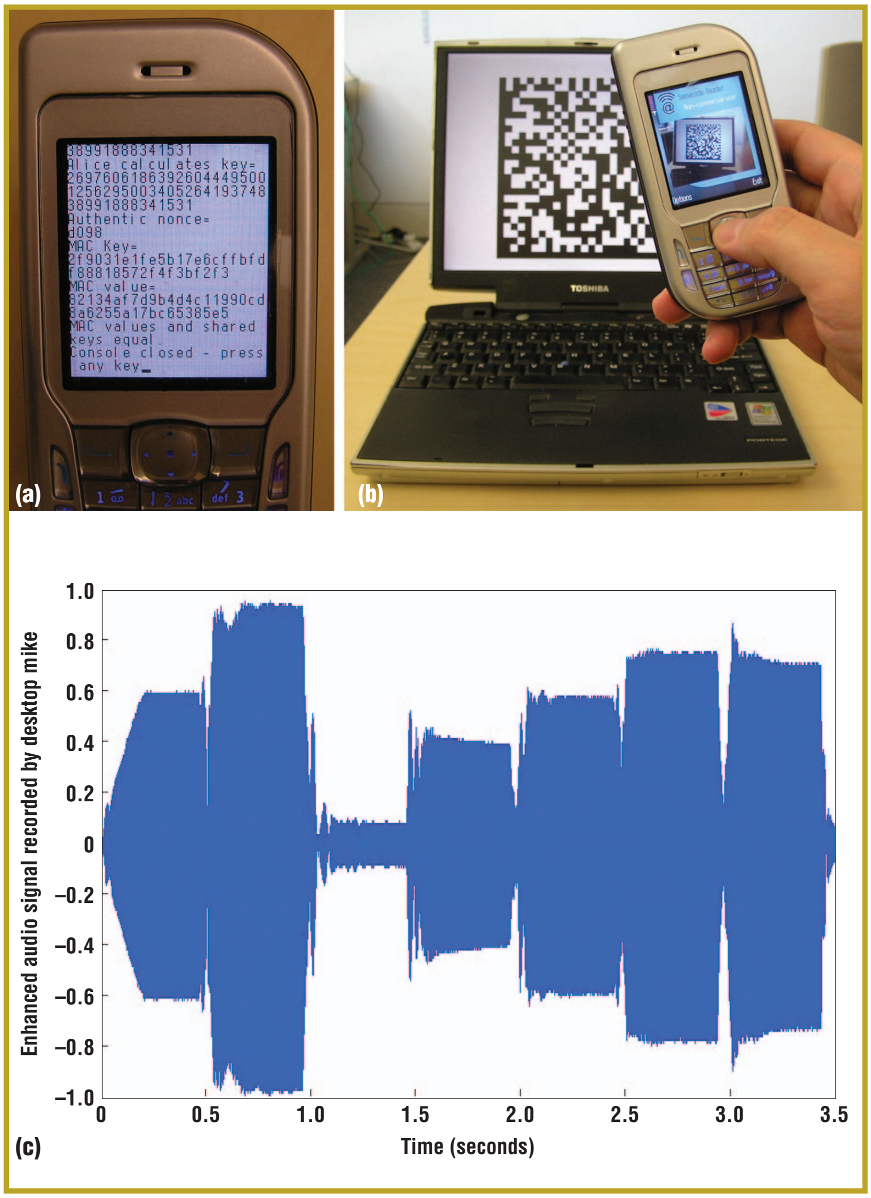
REFERENCES

1. W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, vol. 22, no. 6, 1976, pp. 644–654.
2. M. Steiner, G. Tsudik, and M. Waidner, "Cliques: A New Approach to Group Key Agreement," *Proc. 18th Int'l Conf. Distributed Computing Systems* (ICDCS 98), IEEE CS Press, 1998, pp. 380–387.
3. O. Pereira and J.-J. Quisquater, "Generic Insecurity of Cliques-Type Authenticated Group Key Agreement Protocols," *Proc. IEEE Computer Security Foundations Workshop* (CSFW 17), IEEE CS Press, 2004, pp. 16–29.
4. D. Dolev and A.C. Yao, "On the Security of Public Key Protocols," *IEEE Trans. Information Theory*, vol. 29, no. 2, 1983, pp. 198–208.
5. F.-L. Wong and F. Stajano, "Multi-Channel Protocols for Group Key Agreement in Arbitrary Topologies," *Proc. 3rd IEEE Int'l Workshop Pervasive Computing and Communication Security* (PerSec 06), IEEE Press, 2006.
6. L.H. Nguyen and A.W. Roscoe, "Efficient Group Authentication Protocols Based on Human Interaction," *Proc. FCS-ARSPA*, 2006, pp. 9–32.
7. J. Valkonen, N. Asokan, and K. Nyberg, "Ad Hoc Security Associations for Groups," *Proc. 3rd European Workshop Security in Ad Hoc and Sensor Networks* (ESAS 06), LNCS 4357, Springer, 2006, pp. 150–164.

recorded the number of successes for each size. We didn't perform usability tests, but we expect casual users to have a somewhat lower success rate.

The largest frame we could transfer reliably—that is, with at most one failure in 10 trials—was 14 × 14 pixels (12 × 12 = 144 nonborder pixels, of which 80 were used for Reed-Solomon error correction). The frame carried eight code words and took two further seconds to decode on the 6670 after successful acquisition. With the ASCII-based encoding (6 bits per code word) imposed by the API, this meant 48 bits of payload, although in theory 8 code words could carry up to 64 bits of payload. If used as a public-key fingerprint as in protocol 1, 48 bits can still be brute-forced (and 64 wouldn't give that much more margin), but this payload size is quite secure if used according to protocol 2. The largest frame for which at least one transfer out of 10 still worked was 22 × 22 pixels, carrying 180 bits of payload at 6 bits per code word.

With our equipment, the limiting factor wasn't the acquiring phone's camera resolution (1,152 × 864 pixels) or the source display's much lower resolution (176 × 208 pixels). It was the absence of a macro facility: we needed at least 8 cm for the 6670 to focus, but at that distance the 6600's screen was too small for the 6670 to be able to extract high-density Semacodes. We could successfully acquire much larger codes, at distances of 15 to 30 cm, from a laptop's LCD display, as shown in figure 4b. There, the largest frame transferred with at most 1 out of 10 failures was 40 × 40 pixels, carrying 912 bits at 8 bits per code word and taking 10 seconds to decode. Such message capacity is sufficient to defeat brute-force attacks. Most Japanese phones



```
38991888341531
Alice calculates key=
26976061863926044495OO
1256295003405264193748
38991888341531
Authentic nonce=
d098
MAC Key=
2f9031e1fe5b17e6cffbfd
f88818572f4f3bf2f3
MAC value=
82134af7d9b4d4c11990cd
8a6255a17bc65385e5
MAC values and shared
keys equal.
Console closed - press
 any key_
```

Figure 4. Implementation prototypes: (a) an elliptic-curve cryptography port on a Nokia 6670, (b) visual code transfer from laptop to mobile phone using Semacode, and (c) a shaped C-major scale monophonic tune.

today are equipped with macro lenses and automatically decode QR Code (ISO/IEC 18004). We expect that most phones worldwide will eventually become macro capable if visual codes become widespread. This would allow the phone-to-phone transfer of visual codes long enough to act as full public-key fingerprints, defeating the MITM in protocol 1 without requiring our protocol 2.

Melodic audio

We also experimented with transferring a random nonce from one device to another by playing a short monophonic tune. The audio channel provides some integrity—it's hard for Mallory to interfere without the human operator detecting it. Data origin authenticity isn't as strongly guaranteed as by the visual channel (sometimes you can't quite tell where a tune came from), but it's still better than with radio. There's no confidentiality, because anyone in range also hears. A speaker and microphone are cheaper than an LCD and camera and, especially on the source side, smaller; this might make them more suitable for certain ubicomp gadgets. It's also harder for the operator to miss the fact that a transmission is taking place, which might be good for security but bad for usability. We tried to address the latter point by making the tunes more pleasant-sounding.

Our prototype algorithms generate 3.5-second monophonic tunes, encoding values in the notes' pitch. We didn't explore the limits of the transmission range, but with commodity hardware (external PC speakers and mobile-phone microphones), we repeatedly achieved zero symbol errors over a room-scale distance of two meters (no failures over 10 trials, without error correction).

We sought musical guidance and developed three monophony (music with a single unaccompanied melodic line) generation algorithms: the first randomly chose notes from an octave, the second chose notes only from the C-major scale, and the third restricted large pitch changes between consecutive notes. Figure 4c shows the waveform of a generated audio signal received by the PC microphone. In informal usability experiments with 14 human listeners aged 20 to 35, the algorithms received "pleasantness" scores averaging 3.1, 4.7, and 5.6, respectively, on a scale of 1 (worst) to 10 (best), revealing clear preferences for the third algorithm. This also illustrates a security-versus-usability trade-off because, for the same monophony length, our most listener-friendly scheme gives an entropy of around 15 bits per tune, while the least listener-friendly scheme gives 25. There are opportunities for further research in determining algorithms for generating high-entropy but pleasant melodies, as human perception of pleasantness is quite subjective and subject to cultural influences.

Finally, no matter which auxiliary

channel is used, designers shouldn't make excessive demands on the human user in terms of memorability, concentration, or sensory acuity.

We encourage you to explore ways to apply multichannel protocols to new problems. This area could become a fertile new field for security-protocol research.

the AUTHORS

Ford Long Wong recently submitted his PhD dissertation in computer science at the University of Cambridge. His research interests include cryptographic protocols, authentication, privacy, and ubicomp systems. He received his MEng in electrical and electronic engineering from Imperial College London and his MSc in statistics from the National University of Singapore. He's a member of the IEEE and the International Association for Cryptologic Research. Contact him at the Univ. of Cambridge, Computer Laboratory, William Gates Bldg., 15 JJ Thomson Ave., Cambridge CB3 0FD, UK; ford-long.wong@cl.cam.ac.uk.

Frank Stajano is a senior lecturer in the Computer Laboratory of the University of Cambridge, where he received his PhD in computer science. His research interests include computer security, ubiquitous computing, and privacy in the electronic society, and he's the author of *Security for Ubiquitous Computing* (Wiley, 2002). He was elected a Toshiba Fellow in 2000. Contact him at the Univ. of Cambridge, Computer Laboratory, William Gates Bldg., 15 JJ Thomson Ave., Cambridge CB3 0FD, UK; frank.stajano@cl.cam.ac.uk; www.cl.cam.ac.uk/~fms27.

ACKNOWLEDGMENTS

We thank our colleagues in the Security Group and the Digital Technology Group at the University of Cambridge's Computer Laboratory for providing a stimulating environment in which to conduct this research. We particularly thank Oistein Andersen, Ross Anderson, Saar Drimer, Mark Lomas, and the anonymous reviewers for their useful comments.

REFERENCES

1. F. Stajano and R. Anderson, "The Resurrecting Duckling—Security Issues for Ad Hoc Wireless Networks," *Proc. Security Protocols Workshop*, LNCS 1796, Springer, 1999, pp. 172–182.
2. F.-L. Wong and F. Stajano, "Multi-Channel Protocols," *Proc. 13th Int'l Workshop Security Protocols*, LNCS 4631, Springer, 2005.
3. M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," *Comm. ACM*, vol. 36, no. 3, 1993, pp. 75–84.
4. W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, vol. 22, no. 6, 1976, pp. 644–654.
5. D. Dolev and A.C. Yao, "On the Security of Public Key Protocols," *IEEE Trans. Information Theory*, vol. 29, no. 2, 1983, pp. 198–208.
6. S. Laur and K. Nyberg, "Efficient Mutual Data Authentication Using Manually Authenticated Strings," *Proc. 5th Int'l Conf. Cryptology and Network Security* (CANS 06), LNCS 4301, Springer, 2006, pp. 90–107.