Exploiting Context Histories in Setting up an e-Home

Johannes Helander
Microsoft Research
1 Microsoft Way
Redmond, WA 98007 USA
+1 425 882 8080
jvh@microsoft.com

ABSTRACT
Turning a home into a seamlessly integrated, yet secure environment, without excessive cost, presents a number of challenges. This paper attempts to draw various solutions together. What must be done for security, can also be exploited in making configuration less tedious.

The home environment is augmented by low-cost invisible computers that let everyday objects communicate and integrate. Embedded XML web services are used as a generic substrate for exchanging information between all classes of devices: simple light switches to complex personal computers. Solid cryptography and a touch based trust establishment protocol allow setting up a secure home completely independently. Finally the human interaction context history is used to heuristically determine how the different devices should interact.

Keywords
Home automation, XML web services, embedded systems security, context histories.

INTRODUCTION

An automated home can make our lives more comfortable and easier, perhaps lengthening the time the aging population can stay independent. Countless homes are already filled with personal computers, music systems, appliances, light dimmers, and security systems. If all these systems could work together the utility of all the devices would improve.

There are two main problems in the status quo, however. 1) The systems are difficult to set up and it is impossible to make them work together. Beside the physical connections, that often could be replaced by wireless connections, the protocols spoken are proprietary and application dependent. The user is relegated to archaic switches and menus to tell the systems what to do. A seamlessly integrated home is currently achievable only for millionaires with professional installation crews. 2) The systems are insecure and compromise the privacy of the inhabitant. By installing automation systems the owner of the home ends up paying for losing control of information pertinent to their sanctum. The use of wireless connections only makes the situation worse.

The author claims that interoperation has to be built right into the basic functionality of the system. Security and privacy is not an afterthought and should not make the system another order of magnitude more complex and difficult to set up. The user experience must be intuitive and natural, preferably completely invisibly result from the unavoidable physical installation without further configuration steps.

This paper proposes a physical touch based functional and trust establishment mechanism that exploits a contextual history of human interaction. Interoperation is achieved through the use of XML Web Services that run on low cost microcontrollers. A public key cryptography based trust manager achieves security without external trust authorities. The trust setup is based in physical touch; as a side effect a context history is created. This context history is exploited to determine functional relations between the devices. A single touch per device is thus all that is needed, and this applies potentially to all the devices in the home

EMBEDDED XML WEB SERVICES

Web services were conceived to solve the e-business interoperation problem. The same problem is very pressing in a home environment. The author has shown in [1] that web services can also perform on low-cost microcontrollers. Entertainment content streaming and privacy add new issues, the use of web services in addressing these issues have been explored in [2] and [3].

TRUST ESTABLISHMENT

A home should be totally controlled by the owners. The owners should also be able to set up their home without outside assistance or authority. This is achieved in [2] by using the Resurrecting Duckling Protocol [5]. It works by defining one device to be the authority—the mother. New devices believe the first other device they see is their mother unless they already have one. The first contact is thus critical and a touch (or proximity) based channel is used. The physical touch signifies a human intent and physical access to the mother device. Each device is identified by a certificate that contains the public key of the device, delivered over the physical touch channel. The certificates are signed by the mother with its private key. The mother certificate is received on the same physical touch channel. All later communication can happen on a regular wireless or other public data link.

There is no need for central certificate authority outside the home. [2] shows how independent authorities can federate to manage mutual partial trust.

CREATION OF THE CONTEXT HISTORY

As the creation of the home trust domain and admitting devices into it involves human interaction, the precise pattern of that interaction can be recorded. A single instance of interaction is a context event.

When new devices are brought into the home, they are touched by the mother, e.g. a smart watch, to make them part of the family. It is also possible to touch a device at other times at will.

REPRESENTING CONTEXT EVENTS

A context event, like any data, is represented as an XML fragment. The fragments are collected by the watch and can be sent to any interested and trusted parties based on event subscription.

```
<interaction time="2005-02-21T18:25:00Z" type="touch">
  <function type="light" subtype="torchier" power="100" unit="watt"/>
  <location="floor" height="1.5" unit="meter"/>
  <watch-buttons>1 2</watch-buttons>
  <contact id="uuid:7796f8ac-ab60-49e5-a2e7-61db77e64096"
           url="http://123.45.67.89/discovery"/>
</interaction>
<interaction time="2005-02-21T18:26:00Z" type="touch">
  <function type="light-switch" subtype="lever" values="on off"/>
  <location="wall" height="2" unit="meter"/>
  <contact id="uuid:1a0e6bd0-806f-4bd8-8e93-c0afd97b1044"
           url="http://234.56.78.90/discovery"/>
 </interaction>
```

The context event contains the type of the device and its functions as well as how to contact it. Beside the time, the event contains the location to the extent known and any available information of what the user did, such as buttons pressed at the time of the interaction. The log is conceptually stored in a distributed data base, where it is available for queries and data mining.

EXPLOITING THE CONTEXT HISTORY

Once a context history is available, the device uses it to determine what it is supposed to do and what other devices it should be associated with. The choice is constrained by the trust domain. Untrusted devices are simply ignored, although federation of trust domains enables limited cross-domain interaction.

The primary source of information is the timestamp in each event. Those events that are close to each other temporally can be assumed to be related. Since the pace of user interaction depends on the speed of the user and on the proximity of the devices, an absolute time difference would be inappropriate. Instead a statistical clustering algorithm is used. First obvious (multiple hours) gaps are used to partition the history into sets. Next every interval is examined and a normal distribution is calculated for a set. The set is then separated into multiple subsets by picking a threshold value for the deviation. The largest deviation gaps are used as cutoff points to separate events into separate subsets. The threshold is progressively lowered and too large sets are separated into two. This is done until every set is of reasonable size (<= 5). Finally it is determined that each of the smaller sets is a separate cluster of related functions.

The clusters are examined for compatible functions such as a light and a light-switch. Compatible functions are then linked together. Simply put, the relative temporal proximity of two compatible devices determines their functional relationship. If no compatible functions are found, the proximity requirement is loosened by backtracking the set splitting until some useful relationships are found.

When location is known, it used together with the temporal proximity to determine overall proximity by mixing temporal and spatial proximity together with heuristically determined weights.

The result of the functional (partitional) clustering is that the light-switch ends up controlling the light given that they were both touched in a reasonably close time span.

If desired, the user can control the process to indicate that the clustering should be split at a given point. For example, pressing button 2 on the watch while touching a device could signal that the current device has nothing to do with the previous devices. This could be done when moving from one room to another without a break in between.

RESULTS

The secure embedded web services and the resurrecting duckling protocol that provides the context events were implemented on a low-end ARM microcontroller [6]. Similar single-chip low-power computers are currently available for roughly $5.

We evaluate the feasibility of the software and the security protocol with measurements. Table 1 shows that the entire software can run on a computer that has 256KB ROM and 32KB RAM. This is available on modern microcontrollers of interest.

Files	ROM	Static RAM	Heap	Stack	Total RAM
BASE	24,676	1,940	2,837		2,777
DRIVERS	11,464	332	896	2,288	3,516
TCP/IP	77,024	3,424	2,648	3,400	9,472
XML	7,860	16	88		104
SOAP	29,504	280	996	4,320	5,596
SECProto	14,180	604	1,848	2,648	5,100
AES	16,532	8			8
RSA	9,784	28	24		52
SHA1	5,436	8			8
C-Library	7,620	12			12
TOTAL	**204,080**	6,652	9,337	12,656	**28,645**

Table 1: Footprint (arm - in bytes) at peak usage

We evaluate whether the solid cryptography is feasible on low-cost devices. Table 2 reveals that the two significant costs are key generation and RSA private key operations. The former only needs to be done, and can be primed at the factory or on the way home. RSA private key operations are needed for certificate signing and key exchange. Each need to be done only once but cannot be done before the device was touched. Luckily the certificate does not have to be signed while touching so the interaction itself is quick.

After touching a device but before two devices can communicate, two RSA private key operations must be done in sequence. This means that it takes almost half a minute before the newly associated devices can communicate to each other, making immediate feedback problematic. Further work will investigate cutting down this delay perhaps driven by the mother device.

The clustering algorithm does not contain any complicated math and can be completely calculated using fixed point integer arithmetic in linear time. This makes it suitable for microcontroller use as compared to more elaborate and sophisticated schemes such as [4] that uses Markov models and Bayesian networks, where the clustering computation itself could exceed the available computational capabilities. The simple clustering algorithm presented here also has the advantage of working with little stored history, yet sufficiently addresses the problem requirements.

FURTHER WORK

User studies would be beneficial in determining the best clustering of events and for tuning the algorithm and for verifying that the results are those expected by most users. Further experimentation with exploiting other known parameters, such as partial locations, might also yield interesting results.

CONCLUSION

It is possible to create an interoperable home automation architecture that is both easy to use and affordable without compromising privacy. Data mining of context histories is a viable way of extracting information from interactions that are already necessary for other reasons. This information, when combined with other known information provide enough context to avoid tedious configuration menus and complicated setup steps. The preliminary work presented in this paper shows that this is possible but user studies are needed to determine whether the heuristic is strong enough to produce results intuitive to most people in variable environments.

REFERENCES

1. Forin, A., Helander, J., Pham, P., and Rajendiran, J.: Component Based Invisible Computing, *3rd IEEE/IEE Real-Time Embedded Systems Workshop* (London, December 2001).
2. Helander, J. and Xiong, Y.: Secure Web Services for Low-cost Devices, *8th IEEE International Symposium on Object-oriented Real-time distributed Computing* (Seattle, May 2005).
3. Helander, J. and Sigurdsson S.: Self-Tuning Planned Actions: Time to Make Real-Time SOAP Real, *8th IEEE International Symposium on Object-oriented Real-time distributed Computing* (Seattle, May 2005).
4. Moore, D, Essa, I., and Hayes, M.: Exploiting Human Actions and Object Context for Recognition Tasks, *7th IEEE International Conference on Computer Vision* (Corfu Greece, 1999).
5. Stajano, F. and Anderson, R. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks *LNCS 1796*, Springer-Verlag, 1999.
6. AT91M63200 Summary, AT91 ARM Thumb MCU, http://www.atmel.com/dyn/resources/prod_documents/1028S.PDF
7. The embedded web services implementation is available at http://research.microsoft.com/invisible/

Algorithm	Operation		Latency on a 25 MHz ARM 7		
			Average	Standard deviation	Per KB
1024-bit RSA	Generate a key pair		290 s	56%	N/A
	Private key	Encrypt/decrypt a block (128 bytes)	12.9 s	<1%	103 s
	Public key	Encrypt/decrypt a block (128 bytes)	0.667 s	<1%	5.34 s
128-bit AES	Encrypt/decrypt a block (16 bytes)		0.254 ms	<1%	16.3 ms
SHA1-HMAC	1024 bytes		79.6 ms	<1%	79.6 ms

Table 2: Speed of cryptographic primitives